Smart Sensors I

Multi-node Wireless Temperature Monitor

Group 164

Mike Schmitz
Joe Schneider
Rio Mascarenhas

ECE 403
November 5, 2002

Abstract

Food safety is a major concern. The number one cause of food related illness is improper cooling of food. To alleviate this, the proposed project is to construct a wireless ad hoc temperature sensor network to monitor the temperature of food cold storage units. The system will display the temperature at all measuring points and issue an alarm if dangerous conditions are encountered. Upon notification, appropriate action can be taken to adjust the temperature of the unit to preserve food quality, or the food will be known to be tainted and can be properly disposed. This report focuses on the background and research that has gone into creating the device, and the description and requirements of the proposed ad hoc temperature collection network. Block diagrams of the required components of the device, which consists of one base unit and several nodes, are provided with detailed descriptions of the requirements and possible components under consideration. Also, the proposed schematics are included and discussed. Lastly, a detailed description of the communication protocol is included along with an analysis of its implementation and usage in the temperature collection network.

Chapter 1
Introduction

1.1 Background

The United States alone experiences somewhere between 6 and 33 million cases of foodborne illnesses each year [1]. Safe food is therefore an important business for the world. One way in which the quality of food is compromised is by storing food in improper temperatures, both during transport and while awaiting consumption at a grocery store or in a domestic refrigerator or freezer. If food is not stored safely, meat or milk can spoil, frozen items can melt, and pathogens have the opportunity to grow to unsafe levels.

On another topic, wireless networks are a relatively recent innovation. These networks allow a number of devices, which for this report will be referred to as "nodes", to communicate with each other over radio, removing the need for bulky cords necessary in earlier days. These networks are now being implemented in large numbers in the form of wireless local area networks (LANs) in the computer industry, as well as in sensor applications, on which this document focuses.

Sensor networks allow the remote collection of data without intervention from the user. This allows the measurement of stimuli in harsh, remote, or hard-to-reach locations. Sensor networks benefit from a wireless implementation in that this removes the need for bulky or expensive wires and allows the mounting of sensors in otherwise unfeasible locations such as on a rotating or flying device.

It has been found that there are generally two topologies used to implement a network of any kind: client-server and peer-peer connection. In a host-client network, multiple clients attach themselves to the host forming what is also known as a "star" topology. This type of network is in general use on the Internet, in which each computer is connected to a hub, switch, or router in the star formation. In a peer-peer network, each node of the network has the ability to connect to any other node, resulting in a "mesh" topology. Both topologies are used depending on application [2].

1.2 Problem Description

Because of the safety and liability issues involved with the storage of food, it is in the food providers' best interest to closely monitor the temperatures in which food has been stored and to quickly correct improper temperatures should they occur. Often the measurement of temperatures is done manually, which is time-consuming, tedious, and has the ability to introduce error into the readings. A more reliable alternative proposed is a network of wireless sensors to monitor these temperatures automatically.

Wireless sensor networks of the past have had a few problems. The range of the radio transmitters has placed a limitation on how close one node must be to another one. The radio transmission power must be balanced so as to have enough power to reach the next node while also using an acceptable amount of battery power to do so. If a wireless network was in a star formation with each node connecting to a host, all nodes have to be in range of the host to communicate with the network at all. Two nodes might be right next to each other but unable to

communicate because one cannot connect to the host. This issue needed to be addressed, and the obvious answer from above is to implement a peer-peer network, in which each node can effectively communicate with every other node within range.

However, implementing a peer-peer network is not as easy as it may look. Imagine a string or line of sensors all within the range of just two other sensors in the network. The node on one end of the string cannot directly communicate with the node on the other end of the line, but it can talk to its neighbor, who can talk to its neighbor, and eventually deliver the message to its intended destination. A network of this sort is called a hopping network because messages can be communicated to nodes in the network out of their range by hopping on those nodes that are close by. This is an easy idea, but complex to implement. Complicated routing tables are necessary for nodes to know if there is a route to another node, and even then the node must know if that route includes itself and to whom it should transmit to continue the sending of the message. Now, imagine that the string of sensors has changed to a blob of sensors, all of which are mobile and may be in the range of many sensors. The problem suddenly just got a lot harder. Add to this the ability for the network to form itself without any outside information (called an "ad hoc" network), as well as the ability to add additional nodes at any time and have them completely assimilated into the network, and the problem suddenly explodes.

1.3 Functionality of the Device

The device proposed in this project will consist of nodes capable of forming themselves into a multi-hop wireless ad hoc sensor network. This device will also have the ability to connect to a temperature sensor and will be capable of returning the readings from the temperature sensor to a base unit acting as a data logger, as well as a handheld device which may roam the network. The temperature sensor will have the proper range and precision to allow a useful and reliable reading of the food storage temperature to be used to ensure safe temperatures were maintained during storage. If a node fails and therefore can no longer act as a router for the remaining nodes in the network, the nodes will be able to compensate by discovering another path. The handheld device will be capable of querying and displaying the temperature readings from a requested node. The base unit will periodically poll the nodes for temperature readings, record the readings, and offer a connection to a computer for downloading data for analysis and record keeping.

1.4 Organization

This report is organized as follows:

- **Chapter 1:** Gives some background, describes the problem, and explains the functionality of the device
- **Chapter 2:** Presents previous work and currently available products, an overview of the requirements and risks, initial block diagrams, and a timeline
- **Chapter 3:** The project is divided into several components with detailed block diagrams, possible options, and chosen direction for each one
- **Appendix:** References

Chapter 2
Previous Work and Design Approach

2.1 Introduction
This chapter describes the previous work that has been performed in the areas of temperature monitoring and ad hoc network construction. The requirements of the project, the involved risks, the basic layout of the network, and the general block diagrams for the base unit and sensor nodes are discussed. A timeline outlining the steps of the project and estimated completion dates is also contained.

2.2 Previous Work
A considerable amount of work has been performed in regards to wireless temperature monitoring and multinode ad hoc networks. The next couple of sections review these technologies as means to lay the groundwork for the project.

2.2.1 Temperature Monitoring
Several products exist on the market today that are designed to monitor the temperature of food. The user-friendliest devices consist of wireless sensors that automatically collect and record the temperature. An example of such a device is the radio-linked temperature, humidity, and event recorder produced by IceSpy [3]. It consists of small wireless sensors that collect and transmit data from within the food cold storage unit to a nearby base station. Through the use of PC software, the collected data can be displayed in graphical form on a computer for easy monitoring of the food temperature. The base unit can also issue alarms depending on temperature requirements preset by the user. All sensor nodes are waterproof and come with a sealed 10-year battery. The normal radio communication range is 200 meters line of sight or a more typical 100 meters in an obstructed environment. The base unit must be in direct communication with all sensor nodes, however the company sells a radio repeater to extend the range. The maximum number of sensor nodes that can be implemented is 32, but any number of base units may be incorporated into the system. The base units are AC powered and are thus not portable. This product could be used as a benchmark for the project's performance, but its lack of an ad hoc network topology, portable base unit, and larger nodal structure makes it unfeasible as a replacement for the requirements of the project.

Another device on the market, offered by Rockwell Scientific, [4] is the HIDRA (Highly Deployable Remote Access). It consists of a wireless data collection network designed to monitor machinery status and performance. Each sensor node, which is either AC or battery powered, is capable of collecting data from up to five sensors detecting conditions such as vibration, temperature, proximity, torque, presence/absence, weight and more. The radio communication system is based upon a reconfigurable, self-organizing multihop network similar to an ad hoc network. The range of each sensor node is approximately 100 meters, and it is capable of communicating with other sensor nodes or the base station directly. The nodes also make use of a 2.4 GHz spread spectrum signal format to minimize radio interference with other devices. This product is also capable of interfacing with Device Net, Ethernet, and other TCP/IP protocols. The major drawbacks of this product are its power consumption levels and price. The peak power draw is one watt, which can make quick use of any battery supply, and at a cost of $1000 per node, large-scale implementation can be financially infeasible.

2.2.2 Ad Hoc Networks

An ad hoc network consists of several short range transmitting nodes. In order to move data from one side of the network to the other, which is often outside the transmitting range of the initiating node, the other nodes in the network operate as routers by forwarding the transmitted data until it reaches the desired destination. This is accomplished by receiving the data, updating it, and rebroadcasting it into the network. Many products incorporate this technology into their network scheme. For example, the HIDRA unit mentioned in the previous section relies on this type of network.

Another product that relies on an ad hoc network communication system is the MICA mote produced by Crossbow [5]. These are low power wireless sensor nodes that are capable of measuring several different conditions including temperature. These devices run on the open source operating system TinyOS, [6] which was developed by UC Berkeley for low power ad hoc communication systems. With some additional hardware and software, these motes could effectively be used to monitor food temperature in a user-friendly method. However, the motes are fairly expensive at a price of approximately $1000 for 3 sensor motes and a base unit.

The communication protocol implemented plays a large role in the success of the ad hoc network. The computer science department of Carnegie Mellon University has conducted a large research project known as Mobile Networking Architectures (Monarch) [7] in order to study the performance and efficiency of several possible protocols. The four major protocols that were evaluated for a wireless LAN ad hoc network are listed below. [8]

- Destination-Sequenced Distance Vector (DSDV)
- Temporally-Ordered Routing Algorithm (TORA)
- Dynamic Source Routing (DSR)
- Ad Hoc On-Demand Distance Vectoring (AODV)

Based on extensive simulations, it was found that each protocol can function very well in a given situation, but they all have certain drawbacks. DSDV performed very predictably, delivering nearly all of the data packets when node mobility was low. However, as node mobility increased, the network performance decreased. TORA operated the worst of the four protocols analyzed, but it did manage to deliver 90 percent of the data packets when the network consisted of 10 to 20 nodes. As the number of nodes was increased to 30, the performance of the protocol was drastically reduced and a significant amount of the data packets were dropped due to a large amount of traffic exhausting the network. DSR performed well at all movement speeds and mobility rates. Its method of source routing did generate a considerable amount of overhead bytes, but this did not have a noticeable adverse effect on the network. AODV performed nearly as well as DSR at all movement speeds and mobility rates and eliminated the source routing overhead. However, its transmission of routing overhead packets necessary for proper operation taxed the network to a greater extent than DSR at the higher mobility rates.

From this study, it appears that DSR or an adaptation of DSR would be the best choice for the project because of its high success rate of data packet delivery. Furthermore, DSR does not rely on periodic transmissions to maintain source routes in the network. This helps to save power,

especially when node movement is kept to a minimum, by decreasing or eliminating the need to find new routes in the network.

2.3 Requirements

The following is an overview of the needs that must be satisfied by the project.

1. The network must constitute a wireless, self-organizing, ad hoc topology.
2. The temperature measurement must be accurate in the range of at least 0º F to 41º F.
3. Data collection must occur often enough to accurately record temperature fluctuations.
4. Radio transmitters must operate through metal objects.
5. Nodes must be resistant to the environmental conditions encountered in a freezer (e.g. presence of moisture and humidity).
6. Nodes should be resistant to minor shocks and impacts.
7. Nodes must be a relatively small size.
8. Power consumption should be kept to a minimum to prolong battery life.
9. Setup and operation of the product must be easy and user friendly.
10. Base unit should be portable.
11. Base unit will have an LCD to display the data.
12. The base unit must interface with a PC for easy storage and analysis of the collected data.
13. The user can set temperature alarm levels.
14. An alarm will sound when the temperature is outside of the set range.
15. The cost must be low enough to allow for an affordable large sized network.

2.4 Proof of Concept / Testing

Three sensor nodes and a portable base unit that can also interface with a computer will be built to demonstrate satisfactory operation of the device. Transmitting temperature data wirelessly between the three nodes and eventually to the base unit will test proper operation. This data will then be displayed on the LCD and computer interface. Setting unacceptable temperature limits will test the alarm function. The ad hoc network topology will be tested by collecting data from a sensor node that is out of direct communication range with the base unit. Limitations in lab facilities and budget constraints will prevent the project's requirements to be fulfilled to the maximum extent. Discrete components will be used in the construction of the device, however smaller surface mount components could be used to reduce the size and power consumption of the sensor nodes and base unit.

2.5 Risk Assessment

There are many inherent risks involved with designing an ad hoc network. Furthermore, the purpose for the network, to monitor food temperatures, raises additional concerns. These risks, along with possible solutions, are explored below.

2.5.1 Crosstalk

Crosstalk is defined as undesirable signals created by the coupling between transmission circuits. This can occur in an ad hoc network if two or more nodes in radio range of one another try to transmit data simultaneously. Their data streams will mix and produce a radio signal that is unusable by the receiver circuit. The primary solution to this it to ensure that if the nodes must operate on the same frequency, they don't transmit data when other nodes are communicating. One way to do this is to wait a random amount of time before transmitting, and then have the

transmission time very small compared to the listening time. Another solution is to have the nodes listen for other transmissions, and then only transmit themselves when it is determined that no other nodes are currently transmitting.

2.5.2 Mobility

The sensor nodes and the portable base unit that constitute the ad hoc network have the possibility to move. This mobility means that certain nodes will not always be in communication range with one another, and data pathways will not always be valid. To prevent the break up of the network, new data pathways must be built when the old pathways are destroyed. This can be accomplished by either rebuilding the pathways before every data transmission, which can generate a large amount of traffic in the network, or the pathways can be rebuilt only when it is necessary, which increases the complexity of the protocol but does save on the number of necessary transmissions.

2.5.3 Overbroadcasting

Each data transmission is intended for a particular receiver, however many nodes may receive the data. Therefore, the nodes must be able to identify which transmissions are intended for them and which they can ignore. A simple way to solve this problem is to assign each node a unique ID number. The transmitted packet will then contain the ID number of the node that is the destination for the transmission. Then any node that receives the data can check the included ID number and then process the data accordingly.

2.5.4 Environmental Concerns

The purpose of the sensor network is to monitor the temperature of refrigerators and freezers to ensure safe levels for food storage. For this to occur, the sensor nodes will be placed inside the cold storage unit and will thus be exposed to harsh conditions such as freezing temperatures and moisture contact. In order to guarantee proper operation, all sensor nodes must be able to withstand the environmental hazards either unprotected or through the assistance of protective packaging.

2.6 General Block Diagrams

The project plan consists of several wireless temperature sensor nodes and a portable base unit to receive and display the data. The basic block diagrams of these devices are shown in Figures 2.6.1 and 2.6.2 respectively. Furthermore, a rough diagram of communication pathways in an ad hoc network is shown in Figure 2.6.3.

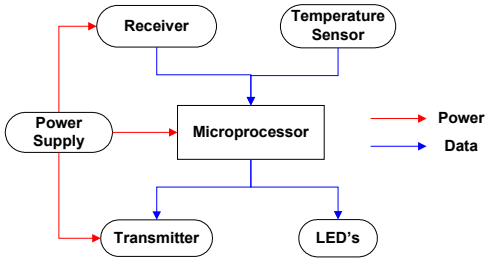


Figure 2.6.1: Sensor node block diagram

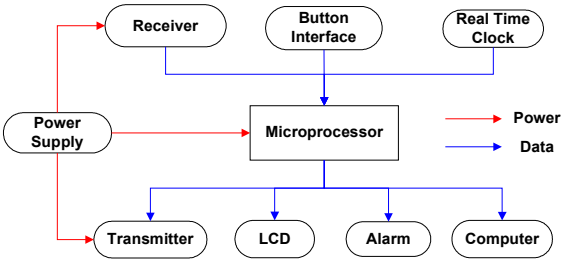


Figure 2.6.2: Base unit block diagram

Microcontroller: Its responsibility is to interpret and execute the protocol used in the ad hoc network topology. It will generate the necessary transmissions needed to build and maintain the network structure as well as receive incoming data and reroute it if necessary. A second microcontroller may be incorporated into the base unit to operate the LCD, buttons, and alarms, thereby leaving the master microcontroller to operate the network.

Transmitter/Receiver: These two components will most likely be incorporated into one transceiver package thereby saving on space and cost. The transmitter will have an effective range of at least 10 feet but not more than 100 feet for power conservation reasons. The radio pair can operate on either AM of FM for this project.

Temperature Sensor: The sensor must be accurate in the range provided in the requirements list. It can either generate an analog or digital signal based on the temperature. If an analog signal is used, an A/D converter must be used in the circuit. This can often times be incorporated directly into the microcontroller.

Real Time Clock: The real time clock chip allows the base unit to record the time at which temperature values were sampled. This will help in data analysis and graphic interpretation on the computer.

Alarm: The alarm will consist of a speaker or buzzer. Its purpose is to alert the user of unacceptable temperature values in the sensor network.

LCD/Button Interface: The LCD and button interface is the direct interface between the base unit and the user. The LCD will display the collected data and the preset temperature alarm levels. The buttons will allow the user to scroll through data on the LCD as well as adjust alarm levels and other settings.

LED's: The LED's on the sensor node will display the status of the node. This could include transmitting, receiving, or standby mode. A switch will be included to turn the LED's off as means to save power.

Power Supply: This will consist of a battery for both the nodes and the base unit. The base unit will also have the option to be plugged in to save on battery power when mobility is not a concern.

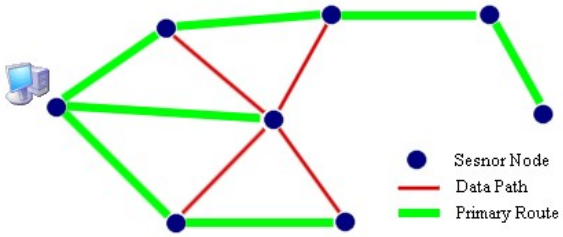


Figure 2.6.3: Ad hoc network topology

The combination of the data paths and the primary routes constitute all of the communication pathways that are possible in the network. Nodes that are far apart have no direct paths to one another because of the limited range of the transmitters. The primary routes will be used for all data transmissions, however if one of these routes were to become inoperable, a new route could be generated by using some of the extra data paths available in the network.

2.7 Timeline

The goal for the completion of this project is December 2002. In order to assure that this deadline will be met, the project has been divided into several tasks. A breakdown of 1st semester is shown in Figure 2.7.1 and 2nd semester is shown in Figure 2.7.2.

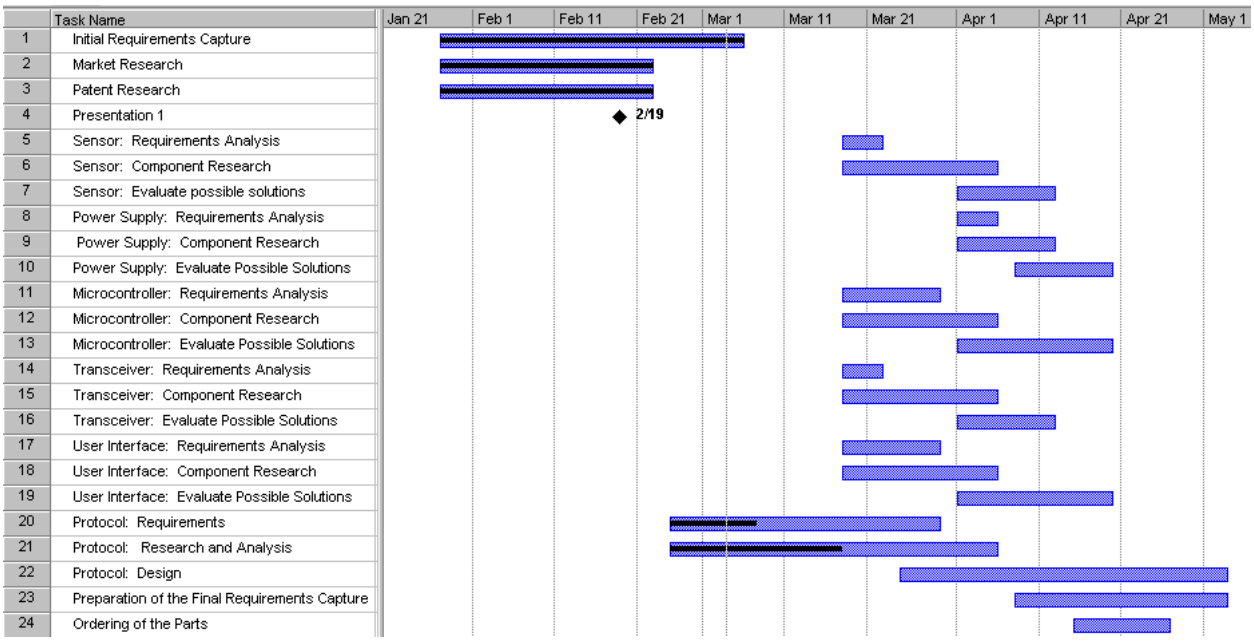


Figure 2.7.1: First semester timeline

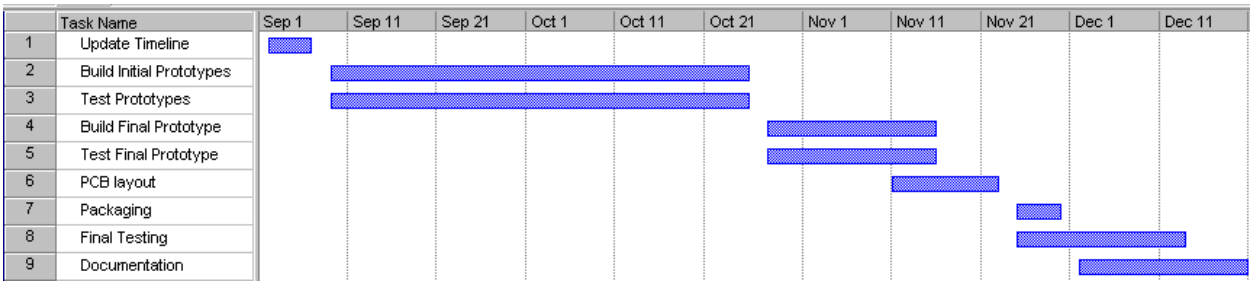


Figure 2.7.2: Second semester timeline

2.8 Conclusion

A breakdown of previous work and commercial products was presented as means to further develop and hone the project requirements. Potential problems that could be encountered were discussed along with several possible solutions. Furthermore, the project was broken down into several component blocks with general descriptions for each. A more detailed component analysis is presented in the following chapter. Lastly, a timeline was presented to ensure project completion in a timely manner.

Chapter 3
Detailed Design

3.1 Introduction
This chapter breaks down the project into manageable components thereby allowing detailed analysis of each. A detailed block diagram indicating the interface structure and main connection points is included in each section. The requirements and several options for consideration are presented for each section, and the ideal option is selected. Several sections also include exact component selection in order to further narrow the component options and requirements.

3.2 Transceiver
The radio transceiver module serves as the lifeline of the network, and therefore, careful consideration and selection of the component is crucial to the speed and functionality of the network.

3.2.1 Detailed Block Diagram
Figure 3.1.1 shows the necessary connections between the microprocessor and the transceiver module. The transmit and receive lines operate on the UART bus and are responsible for sending and receiving the data. CTR0 and CTR1 are control lines and are used to select the transceiver's mode of operation such as transmitting, receiving, or sleep mode. Additional capacitors and resistors may be necessary for the transceiver interface, but this is contingent upon the specific model selected.

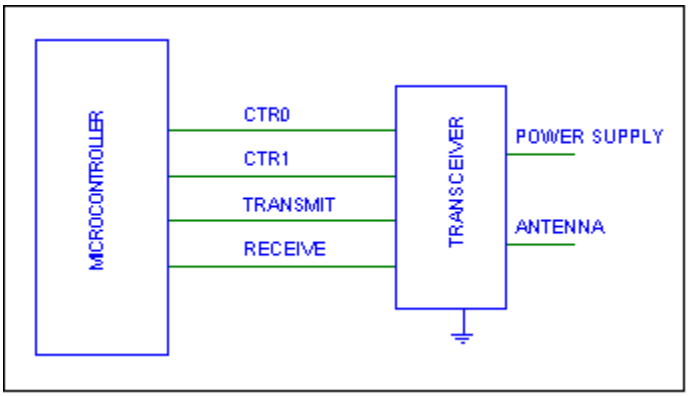


Figure 3.1.1: Transceiver interface

3.2.2 Options Considered
A commercially available transceiver was decided upon due to the complexity and time involved in building one from scratch. The chosen frequency of operation for the radio transceiver is around 900 MHz. This band was selected because it is a widely used commercial band, and therefore, many products operating on this frequency are available. The desired baud rate is relatively low compared to other data-link products. 9600 to 19200 bps is acceptable due to the small amount of transferred data. Furthermore, lower baud rates consume less power. As for operating voltage, 3-volts is required to conform with the other components and the available power supply. Once this criterion is met, transceiver selection will be based on first minimizing

the standby and receiving current draw since that is the primary mode of operation. Finally, transmitting current and switch time delays will be minimized. Additional items to consider are the number of external components needed to interface the transceiver module and the communication range of the transceiver. The minimum required distance is 10 feet and the maximum allowed distance is 100 feet. The range can often times be adjusted by limiting the input voltage of the transmitter.

Three different transceiver modules were analyzed as possible candidates. The two RF Monolithic (RFM) modules were selected because they are similar to the radio transceiver module incorporated in the Crossbow motes. The transceiver manufactured by Linx was considered because it is fairly comparable to the other two transceivers, and it is readily available through the Digikey distributor. A breakdown of the three is shown in Table 3.1.2.

Product	RFM DR3000	RFM DR3000-1	Linx TR-916-SC-S
Data Rate	2.4-19.2 kbps	115.2 kbps	33.6 kbps
Frequency	916 MHz	916 MHz	916 mHz
Modulation Type	OOK	ASK	NA
Supply Voltage	2.7-3.5 V	2.7-3.5 V	2.7-13 V
Operating Temp	-40 to 85 C	-40 to 85 C	0 to 70 C
Sleep Current	5 uA	5 uA	50 uA
Transmit Current	12 mA @ 3 V	12 mA @ 3 V	29 mA @ 5 V
Receiver Current	4.5 mA @ 3V	4.8mA @ 3V	15 mA @ 5 V
Sleep to Receive Switch Time	200 us	20 us	8 ms
Receive to Transmit Time	12 us	12 us	5 ms
Transmit to Receive Time	200 us	20 us	6 ms
Transmitter Output Power	0.75 mW	0.75 mW	NA
Additional Components	2 Resistors	2 Capacitors 1 Resistor	None
Price	Comparable	Comparable	Comparable
Availability	Backorder	Backorder	Digikey

Table 3.1.2: Transceiver options

3.2.3 Option Selected

The Linx module was ruled out because of its high operating and sleep currents. Its temperature range did not match the required range of a freezer either, but ordering industrial, rather than commercial grade transceivers, could have solved this. The RFM DR3000-1 module was at first the favored device because of its fast time delays and high speed, but upon further review, it was realized that obtainable speeds in the network would be much slower in comparison. For this reason, the RFM DR3000 transceiver module was selected. It's baud rate closely matches obtainable speeds in the network, and the slightly less current draw in receiving mode will help to prolong battery life. The timing delays, although slower, are still relatively fast compared to

the operating frequency of the devices. Furthermore, the operating voltage of the module is acceptable for the proposed 3-volt operation level of the sensor nodes.

3.3 Microcontroller

Selection of the microcontroller was initially constrained by the project:

- must consume a very small amount of power to extend battery life
- must be fast enough to effectively manage all peripherals
- must be fast enough to process messages over the network
- must have enough I/O pins to communicate with selected transceiver and temperature sensor for a sensor node, and additionally a real-time-clock, serial communication, LCD communication, and button I/O pins for the base unit

3.3.1 Options Considered

Two basic families of microcontrollers were considered for this project: the PIC mid-range microprocessors and the Atmel AVR 8-bit RISC microprocessors. The decision to narrow the families of microprocessors down to these two was made based on availability and popularity. These two processors are quite different in many ways, although both have desirable features.

Product	PIC mid-range	Atmel AVR 8-bit RISC
Number of Instructions	35	90-130
Max Active Power Consumption	3.8 mA @ 3.0 V	2.2 mA to 5.0 mA @ 3.0V depending on chip selection
Power Consumption when Asleep	5 μA @ 3.0 V	< 1μA @ 3.0 V
Protocol Support	SPI / I^2C / USART Depending on chip selection	SPI / UART Depending on chip selection
Minimum Supply Voltage	2.5 V	1.8 V
Maximum Oscillator Speed at Low Power Consumption	4 MHz	8 MHz
I/O pin combinations available	6, 12, 13, 16, 20, 22, 33, 50, 52	3, 4, 6, 11, 15, 20, 23, 32, 35, 48, 53
Development Costs	Free (Equipment Available)	Programmer is trivial, development software available for free
Maximum Execution Speed	1 MIPS @ 4 MHz	4 MIPS @ 4 MHz

Table 3.3.1: Comparison of microprocessor family features

As can be seen from above, there are quite a few differences to sort through on these two families of devices. First of all, programming will be harder on the Atmel μP because of the large number of instructions, in addition to the fact that none of the group members have experience programming Atmel processors. In this respect, the PIC μPs win hands down.

Also, active power consumption seems to side with the PIC, however the group discussed earlier that the μP will more than likely spend more time sleeping than active, so a lower current consumption in the sleeping mode is preferred, a contest in which the Atmel processor wins.

One huge consideration is the necessity of I^2C, both for a real-time-clock on the base unit, as well as for a temperature sensor on the sensor node. Since the PIC provides I^2C in a hardware solution, the group will not have to be concerned with the complexities of the protocol. However, while the Atmel processors do not have a hardware solution for I^2C support, they do have code provided that will run as an I^2C master, however this is not a very elegant solution and will result in more complex code. The PIC is the winner here.

Looking at minimum supply voltage next, although some of the Atmel processors can run down to 1.8 V, those processors are few. Additionally, a processor which can run on that little voltage is only good if the rest of the peripherals can run down to a 1.8 V supply as well, which, at this time, most are not able to do. Therefore a processor with the capability to run down to 1.8 V is not a considerable requirement, and the requirement becomes one of being able to run on a 3V or below supply, which both processors meet.

To wrap things up, both families have comparable I/O pin configurations, as well development costs. The only cost not listed about the Atmel processors is the time spent learning the system and building and debugging the Atmel programmer. Additionally, the Atmel processor is 4x more powerful than the PIC at the same speed, however 1 MIPS is supposed to be more than enough power for the group's application, so this category becomes moot.

3.3.2 Option Selected

Seeing how two of the members of the group have extensive PIC programming experience, as well as the PIC development equipment being freely available and the PIC parts being available from the department, this group has decided to go with the PIC mid-range microcontroller family. The decision on which particular microcontroller is chosen from this family will be made after the group has decided more of the peripheral I/O line requirements.

3.4 Temperature Sensor

Selection of the temperature sensors was first defined by the project, which defines a few selection criteria:

- must be able to sense ranges in the sub-zero range, preferably down to -10°C
- must be very accurate, since the quality of food and consumers' health depends on it
- must consume very little power, since temperature sensing is an action that will occur often and battery life is to be lengthened as much as possible
- must be readily available and reasonably packaged to both experiment and prototype with

3.4.1 Detailed Block Diagram

The temperature sensor requires four external connections, as shown in Figure 3.4.1 below. The data and clock lines are used by the microprocessor to initiate and retrieve temperature readings. One or both of these lines may require an external pull-up resistor, however this will depend on the particular chip used. The power supply and ground pin are easily supplied by similar supplies for other components.

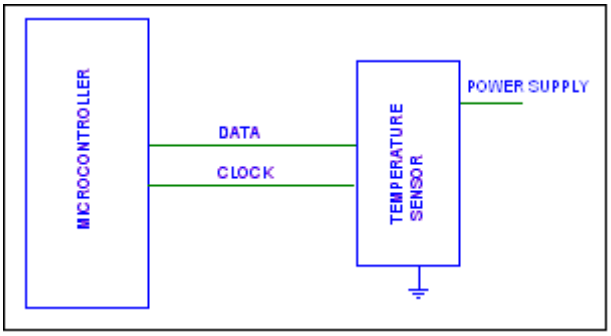


Figure 3.4.1: Temperature sensor interface

3.4.2 Options Considered

The list of possible candidates was narrowed from a large list of temperature sensors to a list of three sensors: the Maxim DS1620, the Maxim DS1621, and the National Semiconductors LM92. All of the candidates met the project's strict requirements well. For the reader's knowledge, the DS1620 was chosen because of its availability in the lab and strong characteristics, while the extremely similar DS1621 and LM92 were chosen strictly based on strong characteristics. A comparison of the chips' features is shown in Table 3.4.1 below.

Product	Maxim DS1620	Maxim DS1621	National LM92
Temp Range	-55°C to +125°C	-55°C to +125°C	-25°C to +150°C
Accuracy in project range	±1.0°C (-10°C to 85°C)	±1.0°C (-10°C to 85°C)	±1.5°C approx (-10°C to 85°C)
Maximum current draw when reading	1 mA	1 mA	625 µA
Typical current draw when reading	1 mA	1 mA	350 µA
Maximum current draw in standby	1 µA	1 µA	5 µA
Price / Qty 1	$6.00	$6.00	$4.05
Availability	Available from limited suppliers	Available from limited suppliers	Available
Communication Protocol	3-wire serial	2-wire serial	I^2C
Typical conversion time	400 ms	400 ms	500 ms

Table 3.4.1: Comparison of features of three considered temperature sensors

As can be seen, the DS162x series is more accurate and has a faster conversion time than the LM92, however the LM92 is cheaper and more available than the DS162x. In addition, there are

some differences in power consumption. The DS162x consume more power than the LM92 when reading the temperature, but less power in standby. This factor could come into play when we decide what the duty cycle of our temperature readings will be. If a relatively fast duty cycle is decided, more time would be spend in the conversion mode and therefore the LM92 would be a more power-conservative choice. However, if the duty cycle is relatively slow, more time would be spent in the standby mode and the DS162x would be the better choice. Seeing as the group had originally discussed around a 5 minute delay between readings, and seeing that the DS162x sensors are superior in both accuracy and speed, a choice was made to go with the DS162x series.

3.4.3 Option Selected

The ensuing question was whether to choose the DS1620 or the DS1621, the only difference being the communication protocol. In favor of the DS1621, it's communication protocol requires only 2 dedicated pins, while the DS1620's 3-wire serial interface requires 3. Both of these protocols are easier to implement than I^2C from scratch, making a serial interface a simpler solution. This having been said, the decision was made to go with the DS1621 chips because fewer pins will be needed to implement a solution.

3.5 Display

The portable base unit requires a display for the user to view the collected temperature data and the corresponding network status. A liquid crystal alpha/numeric character display (LCD) is the most user-friendly and economical solution.

3.5.1 Detailed Block Diagram

Figure 3.5.1 shows the necessary connections needed to interface an LCD incorporating a Hitachi HD44780 controller to the microcontroller. A 4-bit wide bus will be implemented as means to conserve pins on the microcontroller. This is represented by the four data lines. Furthermore, the clock line is necessary to inform the LCD when the incoming data is valid. The select line is used to inform the LCD if the incoming data is an instruction or information to be displayed. The contrast input on the LCD consists of a variable voltage between 0 and 5 volts.

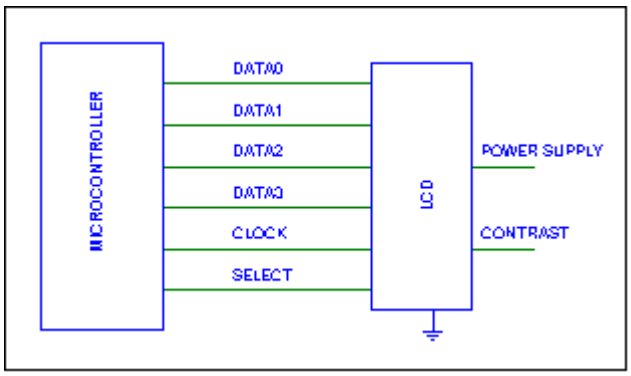


Figure 3.5.1: LCD interface

3.5.2 Options Considered

Few options were considered for the LCD. It was predetermined by the objective of the project that an alpha/numeric character display would suffice. The main option with this is size. The

menu-based display format for the base unit requires a minimum of 4 rows by 16 columns. However, due to the small variance in price between the sizes, a 4 by 20 LCD was selected. This will benefit the user by allowing more information to be displayed simultaneously. Other options to consider are the display controller, backlighting, DC operating characteristics, and price. Table 3.5.2 compares four different LCDs.

Product	DMC-20434U	DMC-20434N	DMC-20434	UC - 20401
Display format	Character display 20x4	Character display 20x4	Character display 20x4	Character display 20x4
Input voltage	4.5 to 5.5 volts	4.5 to 5.5 volts	4.5 to 5.5 volts	5.0 volts only
Current	2 mA	2.3 mA	2.2 mA	NA
Backlighting mechanism	Transmissive	Transmissive	Transmissive	Transmissive
Viewing Area (mm)	25.2 x 76	25.2 x 76	25.2 x 76	60 x 96
Viewing Angle	12 Degrees	12 Degrees	12 Degrees	12 Degrees
Price	$36.40	$42.11	$26.40	$15.95

Table 3.5.2: LCD options

3.5.3 Option Selected

Few differences arise, neglecting cost, in the four LCDs that were compared. For this reason, the UC-20401 LCD was selected for its in-house availability. Furthermore, the LCD incorporates an HD44780 controller. This is beneficial due to relative familiarity of the controller's programming structure and requirements. The LCD uses transmissive backlighting, which is beneficial because it does not consume additional power for an artificial backlight.

3.6 Computer Interface

The computer interface of this project is to be used to move the logged temperature data from the base-unit to a computer for further analysis and a more permanent means of storage. Selection of the computer interface is relatively limited.

3.6.1 Options Considered

The interfaces to be considered are the following: serial communications port, parallel port, USB interface, IrDA interface.

A serial port is relatively easy to build, has been explored in senior design labs before, and fits the application well. While this selection may not be as fast as a USB solution, the amount of data to be moved is rather small, so speed should not be considered an important factor. As long as the data is moved in a reasonable amount of time, which the serial port is able to do, other considerations come first. There are two serial ports available on most computers, which is second in availability only to the USB ports. Both Visual C and Visual Basic have serial communication built-in, which will make programming for a serial solution trivial.

A parallel port solution is probably even easier to build than a serial interface, since the UART of the microcontroller will not have to be utilized. Also, the parallel port is completely

asynchronous, so data can be sent in at whatever rate is necessary for the application instead of being limited by a baud rate. However, parallel ports are very often already in use by a printer, and most computers only have a single parallel port, decreasing the availability of this solution dramatically.

A USB solution would require some daft programming driver programming skills on the part of the group. In addition, the group would need to explore an extra microchip with USB built-in so that USB transfers could be handled by hardware. USB, however, does offer the highest data rate of any of the solutions, along with the ease-of-use associated with USB. In addition, the base-unit could glean power from the USB port while dumping its contents, saving battery power for portable operation.

An IrDA interface would come in handy because no physical connection would need to be made to the computer. Many computers, although often limited to laptops, are equipped with an IrDA port, and so special hardware may be required of the user to connect with this sort of interface if they do not have an IrDA port on their computer. This places the availability of IrDA below all of the other solutions.

3.6.2 Option Selected

After considering the options, a serial connection was decided upon. The serial connection offered the best qualities in terms of availability because of the two ports available on most computers. In addition, serial programming on a microcontroller, while perhaps harder than programming for a parallel port, would be manageable. Despite the fact that the serial port is one of the slower options, it was considered that there was such a small amount of data to be transferred that the time spent to transfer the data would be unnoticeable to the user.

3.7 Real Time Clock

The real time clock chip is incorporated into the base unit as means to generate the date and time of incoming temperature measurements.

3.7.1 Detailed Block Diagram

Figure 3.7.1 shows the necessary connections between the microprocessor and the real time clock chip. The interface shown is a standard 3 wire serial connection. Connections are also shown for the main and backup power supplies. The required crystal must be 32.768 kHz for proper time keeping.

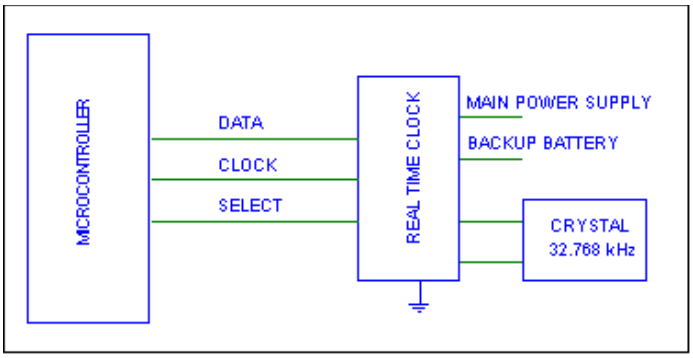


Figure 3.7.1: Real time clock interface

3.7.2 Options Considered

The major difference in real time clocks is the connection to the microprocessor. The desired connection is either I^2C or an SPI interface, but almost any interface, serial or parallel, could be implemented. The ability to connect to a backup battery or other short- term power supply would be desired as well. This would allow the time clock to continue operating even after the main power supply has been removed. Some time clock chips can also trickle charge the backup battery, thereby eliminating the need of periodic replacement. The operating voltage of the chip should allow for either 3 or 5-volts, depending on the chosen operating voltage of the base unit. Furthermore, since the base unit is portable, the smallest possible current draw is required. Other items of consideration are the time clock resolution, with seconds being preferred, and the need of an external oscillator. Three different real time clocks manufactured by Dallas Semiconductor were compared. The details are contained in Table 3.7.2.

Product	DS1302	DS1305	DS1307
Operating Voltage	2.0 – 5.5 V	2.0 – 5.5 V	4.5 – 5.5 V
Active Operating Current	1.28 mA @ 5 V	1.28 mA @ 5 V	1.5 mA @ 5 V
Timekeeping Current	81 uA @ 5 V	81 uA @ 5 V	200 uA @ 5 V
Connection	3 wire synchronous serial interface	Standard 3 wire or SPI interface	2 wire serial interface
Resolution	Seconds	Seconds	Seconds
Battery Backup	Yes	Yes	Yes
Trickle Charger	Yes	Yes	No
Crystal	External	External	External
Date Validity	Up to year 2100	Up to year 2100	Up to year 2100
Alarm	No	Yes	No
Price	$4.16	$5.51	$4.16
Supplier	Digikey	Digikey	Digikey

Table 3.7.2: Real time clock options

3.7.3 Option Selected

All three of these chips could work in the base unit, but they have distinct pitfalls. The DS1307 chip, although comparatively inexpensive, was ruled out because of its relatively narrow operating voltage range and higher current draw. The DS1302 and DS1305 chips are virtually

identical in electrical characteristics, but the major differences are the alarm feature provided by the DS1305, the microprocessor interface, and the price. The DS1302 real time clock chip was selected for the project because of the 3 wire synchronous serial interface. This will take some additional programming to implement, but it allows the I^2C interface to be used for other devices such as the external memory chip. It should also be noted that an external 32.768 kHz crystal is required for operation.

3.8 Memory

The base unit requires additional onboard memory because the processor does not have enough internal memory to store the packet routes of the network, the incoming temperature values, and the corresponding arrival times.

3.8.1 Detailed Block Diagram

Figure 3.8.1 shows the necessary connections between the microprocessor and the memory chip. The interface shown corresponds to the I^2C features of the microprocessor.

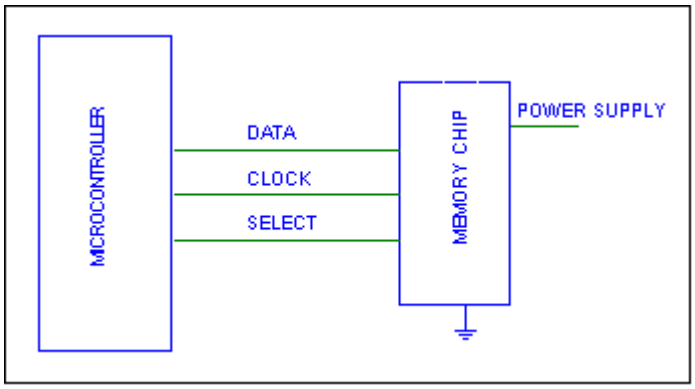


Figure 3.8.1: Memory chip interface

3.8.2 Options Considered

The two main types of read/write memory storage are EEPROM and FLASH memory. While both would be suitable for the project, EEPROM memory was selected for cost reasons. Another consideration is memory size. Size has a direct relationship to the number of nodes that can be controlled by the base unit and the amount of temperature data that can be saved. For this reason, the largest cost viable memory size was chosen. In serial connected chips, this maximum size is 256 Kbits. Four types of EEPROM memory were chosen for comparison and the corresponding characteristics are contained in Table 3.8.1.

Product	24LC256	AT24C256	AT25256	AT28BV256
Connection	I^2C serial connection	2 wire serial	SPI connection	Parallel
Memory size	256 Kbits	256 Kbits	256 Kbits	256 kbits
Operating Voltage	2.5 – 5.5 V	2.5 – 5.5 V	2.7 – 5.5 V	2.7 – 3.6 V
Max Write Current	3 mA @ 5.5 V	3 mA @ 5 V	5 mA @ 5 V	15 mA
Max Read Current	400 uA @ 5.5 V	2 mA @ 5 V	5 mA @ 5 V	15 mA
Standby Current	1 uA @ 5 V	6 uA @ 5.5 V	5 uA @ 5 V	50 uA
Max Clock Frequency	400 kHz	400 kHz	2 MHz	200 ns access time
Price	$1.94	$4.15	$4.79	$10.60
Vendor	Microchip	Atmel	Atmel	Atmel
Supplier	Digikey	Digikey	Digikey	Digikey

Table 3.8.1: EEPROM memory options

3.8.3 Option Selected

Because the base unit is portable, the main concern is power consumption. The 24LC256 chip has the both the lowest operating and standby currents. Furthermore, the I^2C serial connection can be easily implemented by the chosen microprocessor. For theses reasons, and because it is the least expensive and has the largest operating voltage range, the 24LC256 chip produced by Microchip was chosen.

3.9 User Input Interface

The portable base unit necessitates inputs from the user for proper operation. The required inputs include ID numbers for sensor nodes in the network, temperature alarm levels, and the temperature collection rate.

3.9.1 Detailed Block Diagram

The connection format for the input interface will depend on the input option selected. Due to the large number of required inputs, an encoder chip may have to be incorporated in order to decrease the number of required pins on the microcontroller. The block diagram in Figure 3.9.1 shows a generic interface using an encoder chip. The specific interface will be shown in the circuit schematic in Chapter 4.

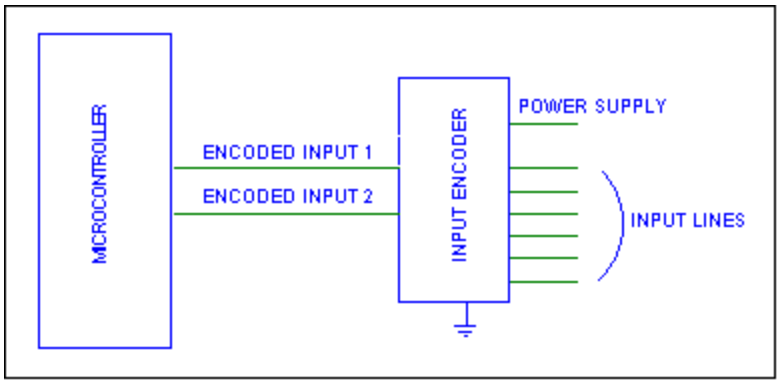


Figure 3.9.1: User input interface

3.9.2 Options Considered

The three main types of inputs include a computer connection, individual buttons, and a numeric keypad. The computer is the easiest input method for the user, but it reduces the portability of the base unit. The user would have to connect to a computer every time a node ID had to be entered or settings had to be adjusted. Individual buttons solve this problem by allowing simple user inputs while away from a computer. Arrow buttons would be extremely useful for a menu based operating system. Lastly, a numeric keypad would allow the user to easily enter node ID numbers and other settings. However, a keypad requires a relatively large amount of space.

3.9.3 Option Selected

To make the unit as user-friendly as possible, an adaptation of all three input methods was chosen. A numeric keypad combined with arrow/select buttons will be used on the base unit. This will allow the user to navigate through a menu-based system and easily enter necessary data. Keypads exist on the market that compactly incorporate numeric buttons along with additional input buttons. The computer connection will also be included to transfer and store collected data on a computer.

3.10 Alarm

When considering options for the alarm, a few selection criteria appear more important than others. The alarm must be audible, so that it is easily heard by those who need to hear it. In the same respect, the alarm must no be so loud as to drain an abnormal amount of power or damage a person's hearing. In addition, simplicity is the key to keeping the code for the alarm simple in the microprocessor code. Some alarms are simply speakers, and would require the microprocessor to send a square or sinusoidal signal to drive the alarm. While this allows flexibility with the frequency of the alarm, it also adds to the complexity of the code. Other alarms are available which just require a constant voltage to drive the alarm. The group decided that the details of the alarm can be left until the second semester, since a suitable alarm will be easy to choose from the many options available. This selection is secondary to the previously mentioned major portions of the base unit.

3.11 Power Supply – Sensor Nodes

Because of the sensor node's unique situation being in small areas for long portions of time, a supply that is as small as possible while still providing enough power and operating at cold temperatures was needed. After reviewing most of the component selections, the group found

that the power supplied to the sensor nodes would only have to be as high as +3 VDC, as all parts had low-power versions that were able to run on low voltages. This allowed an obvious and simple solution of having two AA batteries, each 1.5 VDC, to be used as a power supply. Plastic battery cases to hold two AA batteries were included with the Crossbow Rene motes the group evaluated earlier, and so have been applied before in a viable sensor network. Two AA batteries will offer at least 2000 mAh which translates to a reasonable lifetime before battery replacement for the motes.

3.12 Power Supply – Base Unit

Unlike the situation with the power supply for the sensor nodes in section 3.11, the base unit was not able to be powered by a low voltage +3 VDC supply because of the LCD panel. All LCD panels found needed at least +5 VDC, as well as some even requiring negative voltages. While the group was able to avoid the need for a negative voltage supply, the base unit will still require at least a +5 VDC power supply. The project also called for the base unit to be stationary sometimes and mobile at other times. This prompted the group to consider two powering options for the base unit, one from an AC/DC wall adapter to be plugged into a 120 VAC source when available, and another battery-powered option for mobile use. Therefore, since a case with a battery compartment and LCD hold built in will be easier to find than some other custom-made battery solution and since AA batteries are common, available, light-weight, and relatively cheap, the decision was made to have both a wall adapter solution for stationary operation with a voltage to be determined during schematic design, as well as a 4 AA battery solution for mobile operation. Both supplies will more-than-likely require some sort of regulating circuits, both to regulate the voltages from those input to those required by the circuit, as well as to ensure constant operation during the switch from battery to wall adapter power and vice versa.

3.13 Conclusion

All of the components considered in this project were presented with several options to be considered. After a thorough comparison comparing and contrasting the desirable and less-desirable aspects of each option, an option was selected accompanied by an explanation. This chapter concludes the component consideration and selection stage of the project.

Chapter 4
Protocol Description, Detailed Block Diagrams, and Circuit Schematics

4.1 Introduction

This chapter provides the schematics for the base unit and the nodes, a detailed description of the exact operation and connection of all major circuit components, the format of the communication protocol, and the implementation of the protocol in the wireless ad hoc network.

4.2 Detailed Block Diagrams

Both the nodes and the base unit involved in the network can be broken down in to logical sections, or blocks, which communicate with each other creating a functional unit. A fairly detailed block diagram for both the node and the base unit are presented and described.

4.2.1 Node

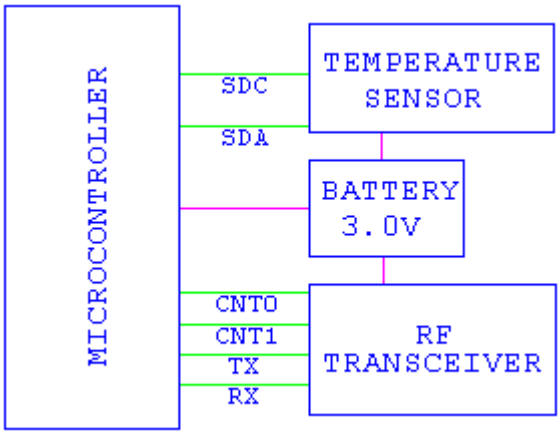


Figure 4.2.1: Node block diagram

The block diagram of a node is shown in Figure 4.2.1. The block on the left represents the microcontroller, a PIC16LF628, and associated discrete components to clock the microcontroller. The oscillator includes a crystal running at 4 MHz and two capacitors tied to ground. The microcontroller is able to communicate with the rest of the network by way of the RF transceiver, a RFM DR3000L. The transceiver is connected to the microprocessor through four lines: CNT0, CNT1, TX, and RX. The CNT0 and CNT1 lines are output only from the microcontroller, and are used to choose the mode of operation of the transceiver. Each CNT line is tied to V_{DD} with a 10kΩ pull-up resistor. Modes of operation for CNT0:CNT1 are given in Table 4.2.1.

CNT0	CNT1	Mode
Low	Low	Sleep (low power)
High	Low	OOK (On-Off Keyed) transmit
Low	High	ASK (Amplitude Shift Keyed) transmit
High	High	Receive

Table 4.2.1: Modes of operation for CNT0:CNT1 pins of DR3000 RF transceiver

The transceiver is used in OOK (On-Off Keyed) mode for transmissions, as this is more power-efficient than the only other transmission option on the transceiver, ASK (Amplitude Shift Keyed) mode. The TX line is an output only line from the microcontroller, and is tied to the transceiver through a 3.3kΩ resistor. The resistor is used to limit the current to the transceiver, which in turn controls the amount of power with which the transceiver transmits. The TX line must be kept at logic low when the transceiver is in receive mode. The RX line is an input only digital line to the microcontroller, representing the demodulated signal received by the transceiver. The only other discrete component used for the transceiver is a 33kΩ resistor which can be optionally connected to pin 8 of the transceiver (the LPF_ADJ pin) to allow for up to 19.2 kbps data bandwidth. Without a resistor attached to pin 8, the transceiver can only transfer data at up to 2400 bps. This resistor changes the speed of the RF link by adjusting the characteristics of the built-in low-pass filter of the transceiver. Further information can be found in the DR3000 data sheet.

In order to simplify the hardware and code complexity involved with communicating wirelessly, the RX line is tied directly to a hardware USART included on the PIC16F628. This allows the microprocessor to sleep until the first byte of a packet is received, and additionally allows the processor to run other code while the rest of the bytes of the packet are clocked in. The TX line is not connected to the hardware USART, however, since the hardware USART idles at a logic high. Letting the TX line idle at logic high causes the transmitter to continuously transmit, which wastes valuable battery power. Instead, a software serial transmitter written to emulate the hardware USART is implemented. The idle high required by the hardware receiver is padded to the beginning and end of all transmissions, fooling the receiving node's hardware receiver into thinking an idle high is being applied.

The temperature sensor, a DS1631, is connected to the microcontroller over a standard I2C two-wire bus. Both lines are tied to V_{DD} by a 10kΩ pull-up resistor. The SDA line is a bi-directional line over which all data is transferred. The SDC line is a clock line controlled by the I2C master, which is the microprocessor. All address pins A0-A2 are tied to ground, resulting in an address of 0x90 after shifting the address left one bit as required by the I2C protocol. The DS1631 can convert temperatures with precision from 9-12 bits, with more bits requiring more time to convert. After the microprocessor requests that the sensor begin converting a temperature, it can determine when the conversion is done by polling the sensor and reading a flag bit. When the flag changes to represent a complete conversion, the microprocessor requests the value of the newly converted temperature, which is then transmitted to the microprocessor.

4.2.2 Base Unit

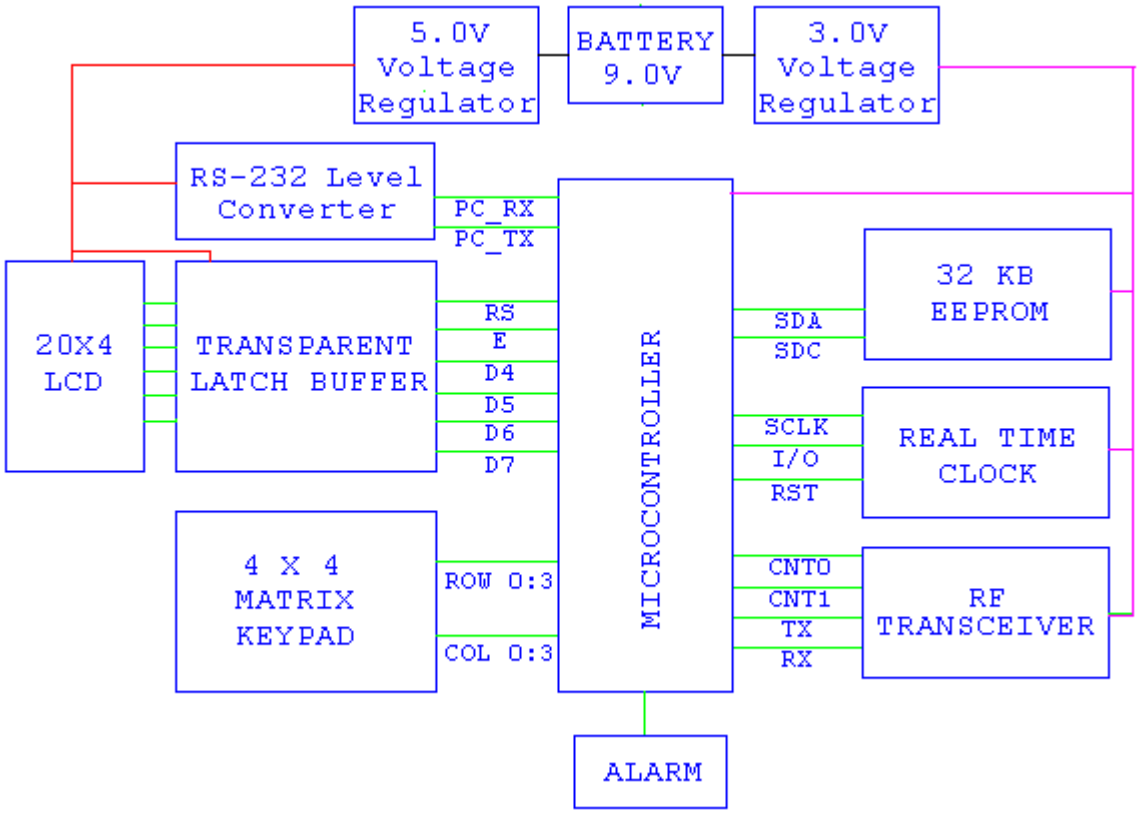


Figure 4.2.2: Base unit block diagram

The base unit block diagram is shown above in Figure 4.2.2. The sections of the base unit block diagram which overlap the node block diagram, namely the microprocessor oscillator components and RF transceiver, will be omitted here. The one difference that should be noted here from the nodes is that the base unit microprocessor is a PIC16LF877 (or PIC18LF452). In addition to the components overlapped by the nodes, the base unit incorporates some components for data storage, accurate time keeping, and data input / output. Furthermore, all components on the base unit operate at 3 V except for the LCD and RS-232 converter. The lower operating voltage helps to reduce power consumption.

In order to store data colleted from the network, the base unit includes a 32 kilobyte EEPROM, the 24LC256. This unit is connected to the microcontroller through the standard I2C two-wire bus. This is the same bus used to interface to the temperature sensors in the nodes, and thus both the SDA and SDC lines in the base unit require the same 10kΩ pull-up resistors that the nodes required. All address lines A0-A2 are tied to ground, giving the EEPROM an I2C address of 0xA0 after shifting the address left one bit as required by the I2C protocol. The SDA and SDC lines are tied to the built-in I2C hardware on the microprocessor. The microprocessor in the base unit, as in the node, acts as the I2C master, providing the serial clock on the output-only SDC pin, and receiving and sending data over the bi-directional data line, SDA.

The real time clock is used to accurately keep the time at which temperature readings are made. These times are then stored along with the temperatures in EEPROM. The real time clock used, the DS1302, requires a 32.768 kHz crystal to keep the time accurately. The time is read and set via a two-wire serial bus. The SCLK line is driven by the microprocessor to provide the serial clock, and the I/O line provides a data path between the RTC (real time clock) and the microprocessor. The RST line is an active-low line which is left at logic low while not accessing the RTC to save battery power. The RST line must be driven to logic high in order to communicate with the RTC.

The matrix keypad, used for data entry in the base unit, is connected to the microprocessor through 8 lines, 4 of which represent the 4 rows of the keypad, and the other 4 of which represent the 4 columns of the keypad. The microprocessor reads the keypad by driving each column line high individually, while keeping all others at logic low. By reading the row lines as inputs when a particular column is driven high, the microprocessor can deduce which key is being pressed. The microprocessor scans through the columns whenever input is expected, waiting for a key press to be detected.

The 20 character x 4 line LCD provides a means of communicating retrieved data from the network to the user. It uses a standard Hitachi 44780 compatible driver IC, and is controlled in the same manner as other LCDs implementing a 44780 driver IC. As many LCD's require, contrast control must be connected to the module with the use of a variable resistor. The LCD module used requires 5V to operate properly, and since the microprocessor, which operates at 3V, needs to drive the lines of the LCD module, a transparent latch buffer was placed between the LCD modules and the microprocessor. The transparent latch buffer, a SN74HCT573, translates the 3.0V logic high of the microprocessor to the 5.0V logic high the LCD module requires. The latch buffer is not required to operate in a transparent mode, but does so since the active-low OE (Output Enable) is tied to ground and the active-high LE (latch-enable) is tied to high.

The RS-232 level converter is a standard MAX232N IC. This chip converts the idle-high TTL serial signals of the microprocessor to industry standard EIA-232 levels for communicating over a serial link with a PC. The MAX232N generates the negative and positive voltages required for EIA-232 by implementing a charge pump, which requires several capacitors. Additional capacitors are added to the circuit to filter the generated voltages, as well as the power supply to the IC. Furthermore, a voltage divider is required on the PC_RX line between the MAX232N and the microprocessor to lower the voltage from 5 V to 3 V to comply with the microprocessor input specifications.

The purpose of the alarm is to alert the user of collected temperatures that fall outside of a user definable range. The actual alarm component has yet to be selected. Therefore, the nature of the input signal is indeterminate. Ideally, an alarm will be selected that requires only a steady rather than oscillating voltage to produce sound, thereby reducing microprocessor consumption.

4.3 Schematics

The proposed base unit and node schematics are contained in the Appendix. The alarm circuit has been omitted because the alarm component has not been chosen. The alarm circuitry will be included in chapter 5.

4.4 Protocol

The network protocol determines the format of all transmissions, the method of communication in the network, and the speed of the system. Therefore, proper protocol selection has a direct impact on the functionality of the network. Several current protocols were analyzed in Chapter 2 as means to determine the best possible solution for the project. However, none of these seamlessly matched the necessities of the proposed network. For this reason, a custom protocol loosely based on the Direct Source Routing (DSR) protocol was designed to meet the distinct needs of the project while keeping power consumption to a minimum.

For data to pass throughout the network, routes from the base unit to each node must be determined. The type of packet responsible for route identification is called Route Discovery. Once useable routes are identified, data can pass through the network without searching for appropriate routes. The packet used for this type of communication is called Route Aware. The general format of a packet is shown in Figure 4.4.1.1. Each portion of the packet will be described in detail.

Packet retransmissions are also discussed. Methods are proposed for handling dropped packets, out of range nodes, and incorrect route information. Finally, cross talk and packet collisions are discussed along with the best methods for prevention.

4.4.1 Protocol Format

Both Route Discovery and Route Aware packets use the same general format shown in Figure 4.4.1.1. Each section of the packet is further dissected.

Figure 4.4.1.1: General protocol format

Preamble –

The preamble is necessary for transmitter/receiver USART synchronization and receiver initialization. The first part of the preamble must consist of no less than 10 high bits, without any start or stop bits accustomed to USART communication. This causes the receiver's USART to reset and therefore perceive the next high to low transition as a start byte. The rest of the preamble consists of 3 bytes with the binary value "01010101." These three bytes are sent with the normal start and stop bytes and are necessary to charge the capacitor in the data slicer of the receiver to a specific level for optimum noise rejection during packet transmission. The three identical and consecutive bytes are also used to key the receiving node on the beginning of a packet transmission.

Address Byte –

The address byte identifies the destination of the packet and the type of packet being transmitted. The most significant bit of the address byte is set if the packet is for Route Discovery, and it is cleared if the packet is for Route Aware. The lower seven bits are used to identify the address of the destination node. Thus, the protocol can accommodate 127 nodes and one base unit. If the packet is of type Route Discovery, the address contained in the address byte is the address of ultimate destination for the Route Discovery packet. On the other hand, the address byte identifies the next node in the included route if the packet is of type Route Aware. Figure 4.4.1.2 shows two example address bytes.

Figure 4.4.1.2: Example address bytes

Packet Code Byte –
The packet code byte is necessary to prevent the flooding of the network with Route Discovery packets. By definition, a Route Discovery requires every node receiving a Route Discovery packet to update the packet's route cache with its own address and then retransmit the packet. However, this could create infinite communication loops which would clog the radio frequency and drain power. The packet code prevents this by including a random number in the packet to serve as a packet identifier. Therefore, if a node receives a Route Discovery packet but the packet code is identical to the previous packet code it received, the node will ignore the Route Discovery and not retransmit the packet.

Checksum Byte –
The checksum byte is used for error detection in the packet. The checksum equals the exclusive-or of all bytes in the packet excluding the preamble and the checksum byte itself. The value of the byte is calculated by the transmitter, sent with the packet, recalculated at the receiver, and compared. If the checksums do not match, an error occurred in communication, and the receiver will reject the entire packet.

Length Byte –
The length byte is necessary for the receiver to identify the number of routing addresses and data bytes contained in the packet, and the total length of the packet. The highest four bits of the length byte indicate the number of data bytes in the packet, anywhere from 0 to 15. Likewise, the lowest four bits of the length byte indicate the number of addresses contain in the route cache of the packet, anywhere from 0 to 15. With this information, the total length of the packet can be calculated as the sum of the address bytes in the route cache, the data bytes, the 4 overhead bytes, and the three bytes in the preamble.

Route Bytes –
The route bytes comprise the packet's route cache. During a Route Discovery, the route cache is where a relaying node would add its address, and thus build a route structure for the receiving node. For a Route Aware, the route cache contains the addresses of the nodes the packet must follow to reach its destination.

Data Bytes –

The data bytes are the reason for the packet in the first place, to transfer information. Any type of information can be stored in this location, up to a maximum of 15 bytes.

4.4.2 Route Discovery

When the base unit needs to initiate communication with a given node, it first searches its memory for a suitable route. If no route is found or if the stored route no longer works due to an ever-changing network configuration, the base unit must generate and transmit a Route Discovery packet. The purpose of this packet is to travel throughout the entire network, collecting route data in the process, until it reaches the intended node. Upon reception, the desired node will analyze the collected route data and generate a Route Aware packet using the collected route data and, at the same time, append any necessary data. This packet will propagate through the network, passing only through the nodes specified in the route data, until it reaches the base unit. The base unit can then analyze the received Route Aware packet and store the included route data in memory. The base unit now has a working route to the given node. Any included data in the packet is also available for the base unit's use.

Route information is collected by the Route Discovery by having every node receiving the packet modify it. When a node receives the Route Discovery, it first checks the address byte to see if it is the intended destination of the packet. If it is not, the node must update the route bytes. It does this by inserting its own address before the first route byte in the packet and incrementing the length byte.

4.4.3 Route Aware

Once the base unit acquires and stores to memory a working route to a node through the use of a Route Discovery, it can communicate to the node using the other type of packet called Route Aware. The benefit to using a Route Aware is that the packet already contains the route information necessary to directly deliver the packet to the intended node, whereas a Route Discovery would be transmitted throughout the entire network. Therefore, a Route Aware requires fewer transmissions thereby conserving power and reducing unnecessary noise in the network.

Likewise, a node can generate a Route Aware packet destined for the base unit. However, since nodes do not store route information in memory, a route back to the base unit must be collected from elsewhere. The easy solution to this is to use the route information contained in the Route Discovery or Route Aware packet received by the node from the base unit.

4.4.4 Packet Retransmissions

When the base unit initiates a Route Discovery, it has no way of knowing if the intended node is in range or even operational. Therefore, in order to establish a route in an acceptable amount of time, the base unit waits a predetermined length of time to receive a Route Aware packet in response to the generated Route Discovery. If no Route Aware is received in the allotted time, another Route Discovery is generated with a new packet code. This process is repeated a finite number of times. If, after several attempts, no Route Aware packet is obtained in reply, the base unit will conclude that the destination node is out of range or is not functioning properly. The base unit will cease all attempts to build a route to the node. However, because the node may later come in range or online, the base unit can try to establish a route again at a later time.

The same is true for Route Aware packets. When the base unit generates a Route Aware packet, it has no way of directly knowing if the designated node received the packet. Therefore, when the desired node receives the Route Aware packet, it must generate another Route Aware packet in response destined for the base unit. When the base unit receives this packet in response, it knows the node successfully received the original packet. Repeated attempts by the base unit similar to the methods used for Route Discovery to establish communication are implemented as well. Furthermore, because packets can become lost or corrupted anywhere in the network, the intended node may actually receive the Route Aware packet without the base unit receiving a Route Aware packet in response. Therefore, a drawback of this technique is that the node may receive the packet more than once.

If a Route Aware packet it being used by the base unit to establish communication with a node and it proves unsuccessful, the problem may be incorrect routing information. To solve this problem, the base unit can revert to using a Route Discovery packet to build a new route to the node.

4.4.5 Packet Collisions

To prevent packet collisions and cross talk in the network, especially during Route Discovery, steps need to be taken to prevent nearby nodes from transmitting at the same time. Two methods have been implemented to address the issue. One method is to have the node wait a random amount of time before transmitting the packet. Therefore, when several adjacent nodes receive a packet simultaneously, they won't all retransmit it at the exact same time. The other method is to listen to the transmitter radio frequency for interference. Once the node determines that no other nodes are transmitting in the area and no other noise is present, it can begin transmitting the packet. Furthermore, the higher the baud rate, the less time it takes to transmit a packet. This in itself will help decrease the amount of collisions.

4.4.6 Example Packet Communication

To further illustrate communication between nodes, a few examples of packet communication are provided below. These examples show the content of a packet at each hop along the network for a Route Discovery and a Route Aware packet. Packets shown are missing the preamble, which is still necessary, but omitted here for brevity.

The following diagrams follow the same convention set forth in Figure 2.6.3 of this document, where thicker green lines represent primary paths for data travel, thinner red lines represent a secondary data path, and blue dots represent a node. These figures extend the previous legend to include packets, which are presented in a mono-spaced font and given in standard hex representation of each byte, each byte being separated by a single space. Each packet is drawn above the node which is transmitting, and the direction the packet travels across data paths is shown by arrows for nodes within range of the transmitting node. PC is used as a generic packet code as described above, and in cases where transmission requires the use of two packet codes, the two different codes are identified by a box around the second PC. Similarly, a generic checksum byte is given just as CS, as the checksum will vary depending on the packet code and other included data bytes.

These examples assume ideal network operation in which no collisions occur, i.e. no nodes attempt to transmit at the same time. It is important the reader realizes this idealized communication is valid for most transmissions, but may fail under certain circumstances.

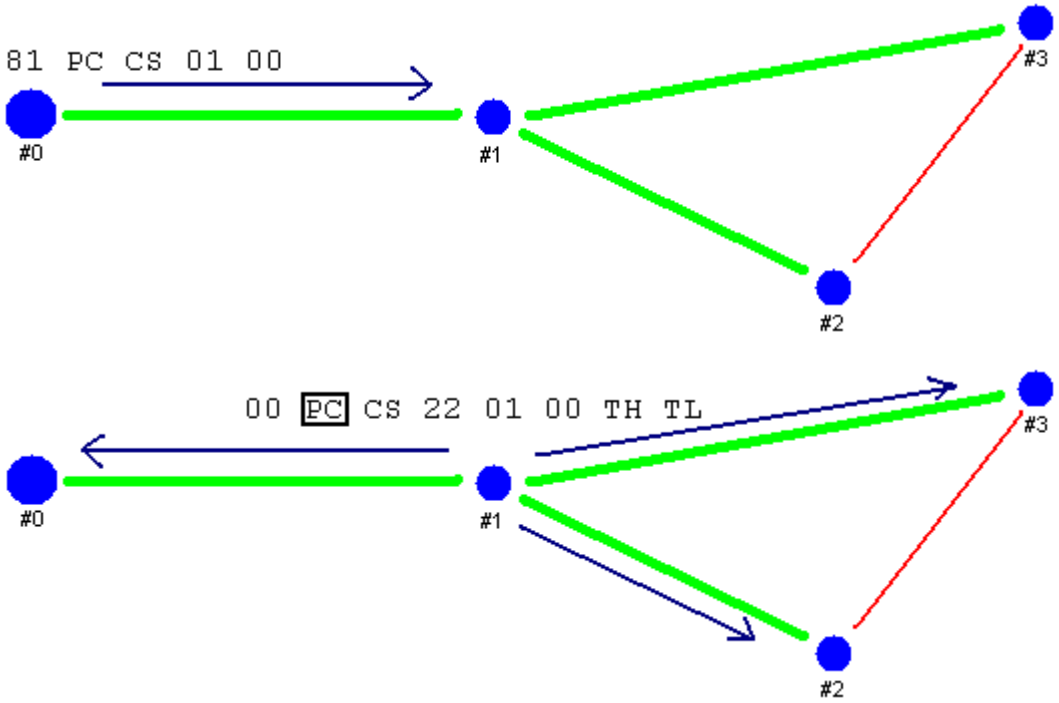


Fig 4.4.6.1: Example Route Discovery from Node #0 to Node #1

Route Discovery from Node #0 to Node #1 –
In Figure 4.4.6.1 a route discovery packet is shown being sent from node #0 to node #1. The Route Discovery is identified by the 8 in the first nibble of the address byte. The 1 in the second nibble of the address byte represents the final destination of the Route Discovery, in this case, node #1. The fourth byte shows the byte has 0 data bytes and 1 route byte. Finally, the fifth packet represents the route that has been accumulated so far, only consisting of node #0.

Node #1 receives this packet and sees that the packet is indeed destined for itself. Seeing this, it adds itself to the route, changes the header of the packet to represent a route aware packet destined for the node originating the route discovery, generates a new packet code, and appends the temperature data to the end of the packet. Both nodes #3 and #2 receive this packet, but quickly ignore it because the header represents a route aware packet which is not destined for either of them. Node #0, on the other hand, receives this packet, stores the route data contained within, and retrieves the temperature data stored within. At this point, the transmission is complete.

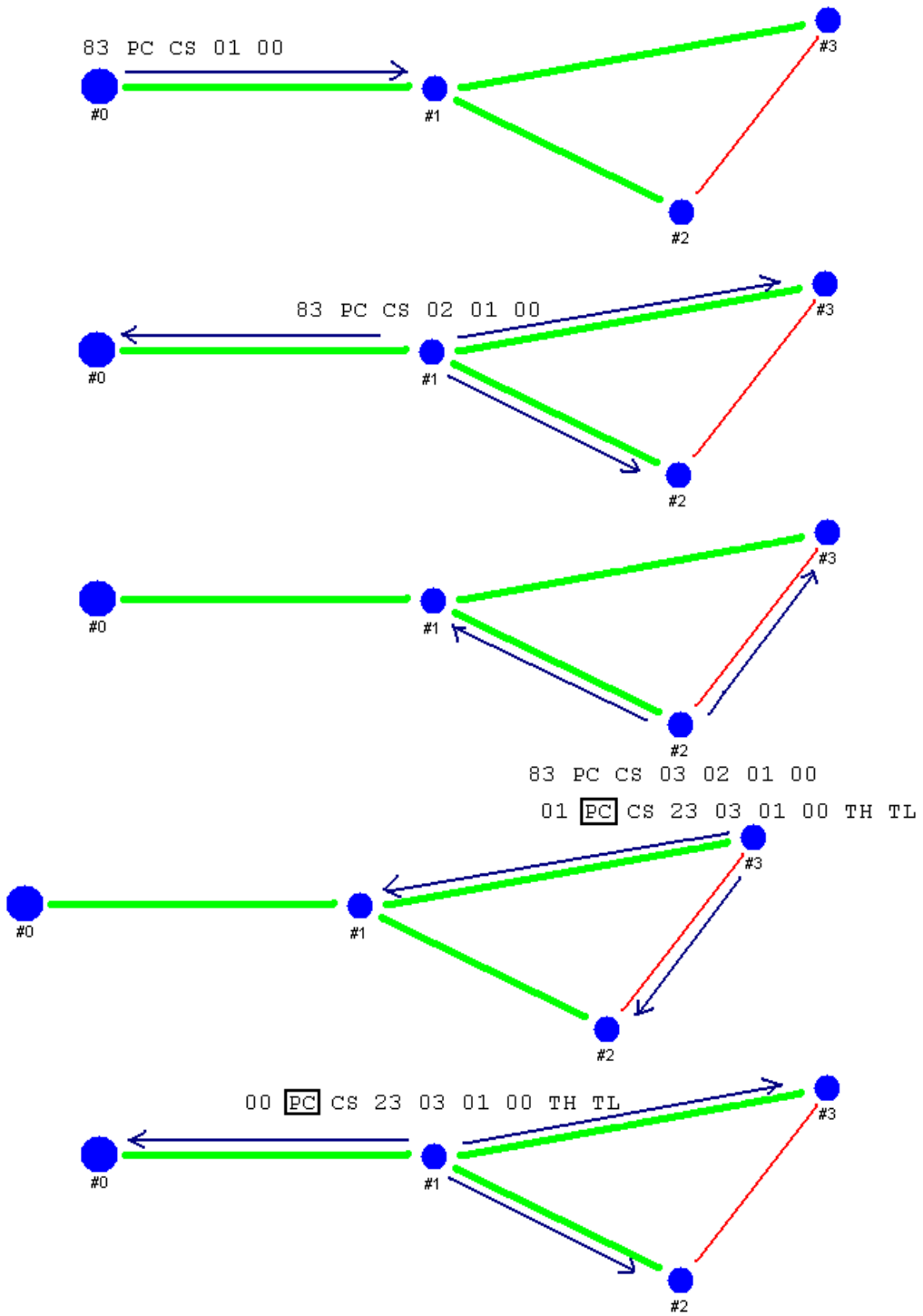


Figure 4.4.6.2: Example Route Discovery from Node #0 to Node #3

Route Discovery from Node #0 to Node #3 –
Figure 4.4.6.2 is quite a bit more complex than the previous example shown in Figure 4.4.6.1 since it adds both multiple hops and multiple paths to the example. In this example, a packet is first sent from node #0 to node #1 representing a route discovery for node #3, shown by the first

byte 83. As in the previous example, there is only 1 route byte in this packet, representing the node originating the route discovery, node #0. Node #1 receives this packet, sees that it is a Route Discovery not destined for it, adds its address to the route information, and retransmits the packet with augmented route information. Node #0 receives this information, and ignores it because it recognizes the packet code as the same code it recently sent to node #1. Nodes #2 and #3, however, see this packet and act upon it differently. Node #2 sees the packet is not destined for itself and begins to change the packet to include itself as a hop in the route. Meanwhile, node #3 has received #1's transmission, realizes this is a route discovery destined for itself, creates a new Route Aware packet containing the newly acquired route information, and begins a temperature conversion to include with the packet in the data bytes. More than likely, this temperature conversion will take longer than node #2 will take to retransmit, and node #2's transmission will not be heard by node #3. However, node #1 will hear node #2 and will quickly ignore this packet, as it recognizes the packet code. Eventually, node #3 will finish converting the temperature and will finally send the Route Aware packet specifically destined for node #1 with a new packet code. This packet will be ignored by node #2 because it sees the packet is destined for node #1 only, represented by the second nibble in the address byte. However, node #1 will see the packet as a Route Aware packet destined for itself and will pass it on using the route information contained within the packet, setting the destination of the packet to node #0, the next node in the route. Nodes #2 and #3 will hear this packet, but will ignore it because they recognize the packet code. Node #0 will receive this packet, see it is destined for itself, store the route to memory for future use, and receive the temperature data contained within the data bytes. With this reception, the transmission is complete.

Route Aware from Node #0 to Node #2 –
Figure 4.4.6.3 demonstrates a Route Aware communication from node #0 to node #2. This example assumes that node #0 has previously acquired the correct route to node #2, and is now utilizing that information to create and send a Route Aware packet. In this case, node #0 creates a Route Aware packet destined for the first node in the route, node #1. This is shown in the first byte, 01. Node #1 sees this route aware packet destined for it and retransmits it, changing the destination to the next node in the route, which is node #2. Node #3 sees the packet and ignores it since it is not destined for itself. However, node #2 sees that the packet is destined for itself. Node #2 then generates a Route Aware in reply by flipping the route information in the packet, setting the destination of the packet to the first node in the return route, node #1. Node #2 also generates a new packet code for the packet and tacks on its temperature data to the data byte portion of the packet. Node #3 hears the transmission from node #2, but quickly ignores it because the packet is not destined for node #3. Node #1 sees the packet is destined for itself, changes the address byte to reflect the next node in the route, and retransmits it. Nodes #3 and #2 hear this transmission, and ignore it because the transmission is not destined for either of them. Node #0 hears the transmission and sees it is destined for itself. It can then use the data bytes of the packet to discern the temperature at node #2. With this, the transmission is complete.

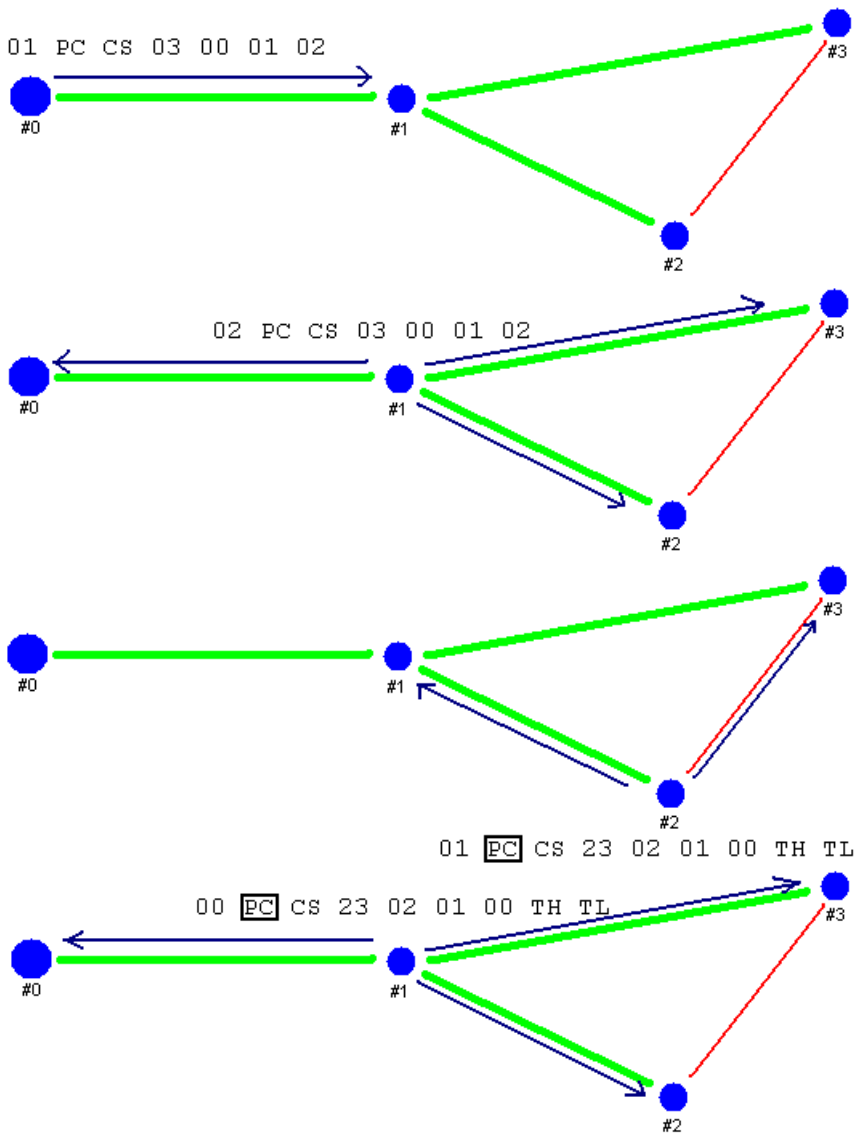


Figure 4.4.6.3: Example Route Aware from Node #0 to Node #2

4.5 Protocol Implementation in the wireless temperature network

The constructed temperature network uses the ad hoc protocol in the exact format provided. When the base unit is instructed to collect a temperature from a given node, it first searches the microprocessor's internal EEPROM memory for a route to the node. The routes are stored in EEPROM so the base unit does not waste power and time rebuilding routes every time the base's power is cycled. If the base unit finds a route in EEPROM for the given node, it constructs a Route Aware packet. Otherwise, it builds a Route Discovery packet. Once the necessary packet is created, the base unit waits for communication and interference on the network to cease by monitoring the receiver for any incoming data. At that point, the base unit will transmit the packet once, and wait for a preset amount of time, currently approximately one second, for a response from the given node. During this waiting time, the base unit will ignore all other packets it receives except for packets from the node it transmitted to.

If the base unit does not receive a reply form the node after the first attempt, the base unit will transmit another packet of the same type containing a new packet code. The base unit will once again wait for approximately one second for a response from the node. If communication fails again, the base unit will transmit a Route Discovery for the given node as the third attempt. Furthermore, if a route for the node was stored in EEPROM, it will be erased because it has proved ineffective in establishing communication. If communication fails for the third time, a Route Discovery will be sent one last time in attempt to reach the node. If a response is not received, the base unit will halt all attempts and deem the node a failure until further notice. Therefore, the maximum number of attempts at communication the base unit will make is four.

If or when a response is received, the base unit confirms the packet is correct by checking the packet code, checksum, and that only two data bytes are included. If the packet checks out, the base unit copies the routing information contained in the packet to the microprocessor's EEPROM for use next time communication with the same node is desired. The two data bytes, which contain the temperature from the node, are either converted and displayed on the LCD for viewing or copied to the external EEPROM on the base unit along with the date and time of reception provided by the real time clock chip for later access. The base allows large chunks of data to be displayed on the LCD or downloaded to a personal computer for analysis.

4.6 Conclusion

The inner workings and intermediate signals of the designed base unit and nodes were provided along with the proposed schematics for each. Furthermore, the format of the protocol, examples of protocol usage, and actual implementation of the protocol in the project was discussed.

Chapter 5

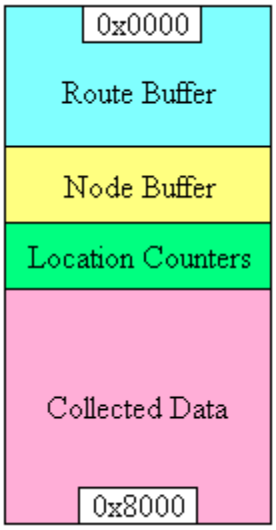
Memory Management, User Interface, Schematic Updates, PC software and Protocol

5.1 Introduction

This chapter seeks to tie up some of the loose ends in the project. First, the methods of managing the memory in the base unit will be discussed, the user interface and all options and settings will be considered, all modifications to the schematics will be laid out, and the functionality and protocol for the PC software will be presented.

5.2 Base Unit Memory Management

The base unit contains EEPROM memory in both the microcontroller and external memory chip. These two sections of memory are used to store the collected temperatures from nodes, the corresponding times of collection, the routes to nodes in the network, the nodes from which to collect temperatures, and the minimum and maximum temperature alarm levels. Only the temperature alarm levels are stored in the microcontroller EEPROM, whereas everything else is stored in the external chip. The configuration of the external memory is shown at right. The Route Buffer contains the routes to all nodes the base unit has communicated with. The Node Buffer contains the nodes the base unit should request temperatures from while performing periodic collection. The Location Counters identify where in memory the next temperature reading will be stored and how many readings are stored for each node. The Collected Data block contains the temperatures and corresponding collection times for the nodes identified in the Node Buffer. When the memory is full, incoming temperatures for a node will replace the oldest temperatures for the same node. Therefore, the most recent temperatures are always available.



The external memory configuration is hard coded in the microcontroller for the maximum possible number of nodes and cannot adjust depending on the usage. For example, if less then the maximum number of nodes are polled during periodic temperature collection, part of the Collected Data block will remain empty. This is because part of the block is reserved for the nodes that are not being polled. It would be more efficient to use a dynamic memory configuration to utilize this wasted space, but the code overhead needed to do so would overwhelm the microcontroller.

5.3 Base Unit User Interface

The user interface on the base unit consists of a LCD and keypad. The user operates the base unit by navigating through the menus on the LCD. The main menu is shown below, and each option is described in detail.

```
1. Node temp
2. Start collection
3. View data
4. Connect
```

```
5. Set clock
6. Temp alarm range
```

Node temp –
This option performs a single temperature collection from one node in the network. The user is prompted to enter the desired node's address, a temperature request will be made for the node, and the corresponding temperature will be displayed on the LCD. If communication fails with the node, "Communication Failure" will be displayed on the LCD. The collected temperature will not be stored in memory and is not available for download to a computer.

Start collection –
This option performs periodic temperature collection from a group of nodes. All collected temperatures along with the times of collection will be stored in memory and are available for download to a computer. If any temperature is outside of the temperature alarm range, the alarm will be activated. When this option is selected, the user will be prompted to enter the addresses of all nodes that should be polled, followed by the polling frequency. The polling frequency can range from continuous to once every 256 minutes with a resolution of one second. Once temperature collection begins, the LCD will display only the most recent collected temperature, the address of the node it came from, and the time of collection. However, if a collected temperature falls outside the temperature alarm range, only this temperature will be displayed on the LCD. It will remain on the LCD until replaced by a new temperature that falls outside the temperature alarm range.

View data –
This option allows the user to view on the LCD the temperatures stored in memory during periodic collection. The user will be prompted to enter the address of the desired node, and then the stored temperatures and collection times for that node will be displayed on the LCD in the order of collection beginning with the most recent.

Connect –
This option allows a computer to control the base unit. The computer is able to download the temperatures and times stored in memory, download the routes to nodes in the network, read and set the base unit's clock, and perform single temperature collections from individual nodes.

Set clock –
This option allows the user to read and set the base unit's time and date.

Temp alarm range –
This option allows the user to read and set the minimum and maximum acceptable temperatures. If any collected temperature falls outside this range, the alarm will be activated.

5.4 Schematic Modifications

Several changes were made to the base unit schematic given in chapter four. The four major changes were the addition of a buzzer, a larger capacitor across the power input to the circuit, a larger capacitor across the output of the 3-volt regulator, two power switches, and a PNP transistor that has replaced the voltage divider on the receive line of the RS232 converter. The buzzer that was chosen operates on DC voltage in the range of 3 to 12 volts. A 5-volt supply, rather than the available 9-volts, was chosen for the buzzer because the generated sound was considered efficient. Since the microcontroller could only provide 3 volts, a NPN transistor was

used to switch in the higher voltage. The larger capacitor across the base unit's power supply was needed to prevent the microcontroller from resetting when the power supply was switched between the DC adapter and battery. The smaller capacitor could not supply enough power to maintain proper circuit operation during the short power loss caused by supply switching. The larger capacitor across the output of the 3-volt regulator was required to prevent the real time clock from accidentally resetting during base unit power up. At power up, enough current is drawn by the circuit to momentarily cause the 3-volt regulator's output voltage to drop, sometimes into unacceptable ranges. The two switches are part of one double pole switch. They are used to turn the circuit on and off. One switch controls the 5-volt supply to all the necessary components, and the other switch controls the 3-volt supply to the remaining components. The 3-volt power supply for the real time clock (ds1302) is not affected by the switches because it must always remain powered to keep accurate time. The voltage divider was replaced with a transistor to conserve power while performing the identical task as a voltage divider. The remaining changes to the base unit schematic consisted of pin connection changes between the different IC's. These changes were made to simplify the board layout by allowing the IC's to be placed closer together with shorter traces.

The only change made to the node schematic was the addition of an LED and resistor. The LED was added to signify when the node is transmitting. In the shown configuration, the LED operates on negative logic. Both the new base unit and node schematics, dated 11/04/02, are contained in the Appendix.

5.5 PC Software User Interface

The base unit can be connected to a computer for additional functionality and logging capabilities. Using a Java application, the user can collect node temperatures on demand from an intuitive user interface, as well as setup a standard collection, save collected data to a file, print collected data, or even graph temperatures over time.

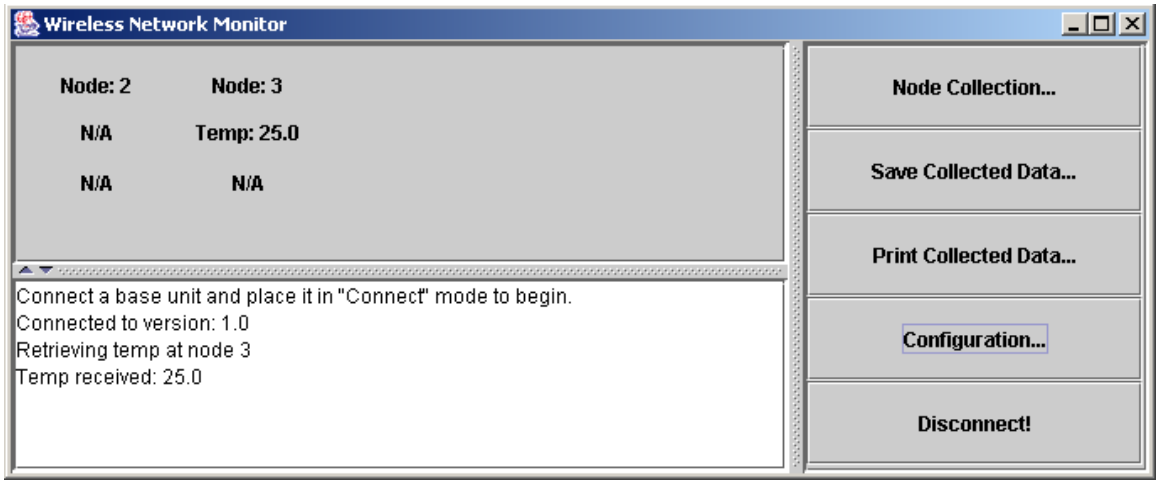


Figure 5.5.1: Wireless Network Monitor

The main network monitor screen is shown in Figure 5.5.1. There are three main panels: the node status pane in the upper-left, the status / log pane in the lower-left, and the control pane on the right.

The node status pane displays the results of the last communication with a particular node. Each node in the network is identified, and relevant information is given. As can be seen in Figure 5.5.1, node 2 has not been queried yet and so does not have any information available. Node 3, however, has been queried and has returned a temperature of 25.0 °C. The Java code can be easily modified to include more information with each node. The node status pane also allows the user to query any particular node at any time, by double-clicking on its information. The information in the status pane is then updated when the base unit receives a response from the node.

The status / log pane below describes the current status of the program and records actions taken by the program. The log in Figure 5.5.1 shows that a base unit with firmware version 1.0 was first connected to the computer. Next, node 3 was queried by double-clicking on its information in the Node Status pane, shown by the "Retrieving temp at node 3" message on the next line. The last line in the log shows that a response has been received from the base unit and gives the information contained within the message.

The control pane is where network configuration is setup and where data is manipulated. The first button, "Node Collection…", initiates a node collection similar to the node collection function available on the stand-alone base unit, including a customizable delay for periodic collection. The "Save Collected Data…" button allows a user to download data from the base unit and save it to a file for further analysis or record keeping. "Print Collected Data…" will allow the user to print out a summary or detailed logs of sensor data. "Configuration…" allows the user to add or remove node number from the network. "Disconnect!" stops all current operations to and from the base unit and closes all open log files.

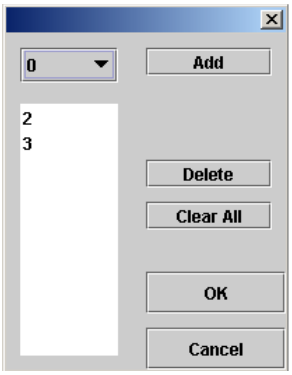


Figure 5.5.2: Configuration dialog

The "Configuration…" button, shown in Figure 5.5.2, provides an easy way to add and remove nodes from the network. From this dialog, users can easily configure the network for the current set of nodes available, or a subset to focus on a particular group of sensors.

Functionality of the base unit PC software is still in its infancy, and will likely be expanded to include the capability to graph previous temperature readings, alert the user if temperatures drift beyond a predefined boundary, provide network route information for network reliability analysis, and set and check the clock. Final operating instructions will be provided later in the user manual.

5.6 Base Unit/Computer Protocol

The PC software that is associated with the base unit requires a standard method of data transfer in order to successfully communicate with the base unit. A custom protocol has been developed for this purpose. It dictates how a connection is initiated, what type of functions can be performed, and the format for all transfers. Before any function can be performed, the PC and base unit must first establish a connection. This is initiated by setting the base unit to connect mode. The remaining connection sequence is shown below.

Connection sequence:

Purpose: To identify the firmware version on the base unit and ensure the PC software supports the firmware version.

Data Direction	Packet	Details
BU -> PC	'sIDV.vX'	V = major version number v = minor version number X = checksum
PC -> BU	'g' or 'b'	g = version supported b = version not supported

Once a connection has been made, the PC software can perform four functions. They are listed and discussed in detail below.

1. Real Time Clock functions
2. Temperature request from single node
3. Sensor route retrieval
4. Previous stored data retrieval

Real Time Clock functions:

Purpose: To set or receive the current time on the real time clock chip on the base unit.

Data Direction	Packet	Details
PC -> BU	'TR'	Read time from RTC chip on BU
BU -> PC	'sMDYHNSCX'	M = month D = day Y = year H = hour N = minute S = second *all variables for this packet are in BCD form X = checksum – not in BCD form

Data Direction	Packet	Details
PC -> BU	‘TS’	Set time of RTC chip on BU
BU -> PC	‘g’	BU ready to receive time
PC -> BU	‘MDYHNSX’	M = month D = day Y = year H = hour N = minute S = second *all variables for this packet are in BCD form X = checksum – not in BCD form
BU -> PC	‘g’ or ‘b’	Acknowledge the set time packet and assert that time has been set. Error with checksum, time will not be set

Sensor Data Packet:

Purpose: To request the temperature from a node. This packet causes the base unit to send a packet out to the sensor and wait for a response from the node.

Data Direction	Packet	Details
PC -> BU	‘QN’	N = node to which packet is sent
BU -> PC	‘sTtX’ or ‘b’	T = upper 8 bits of temp t = lower 8 bits of temp X = checksum Sensor did not respond

Sensor Route Packet:

Purpose: To acquire the route being used to communicate with a particular node in the network.

Data Direction	Packet	Details
PC -> BU	‘RN’	N = node for which route is to be sent
BU -> PC	‘sC(RD)X’	C = number of bytes in route data (RD) = route data X = checksum

Previous Stored Data Retrieval:

Purpose: To retrieve data stored in the EEPROM on the base unit, which would normally have been collected by the base unit before being connected to the computer. When connected to the computer, the computer stores all retrieved temperatures locally, not in EEPROM.

Data Direction	Packet	Details
PC -> BU	'DN'	N = node for which data is to be retrieved
BU -> PC	'sTtMDYHNSX'	T = upper 8 bits of temp t = lower 8 bits of temp M = month D = day Y = year H = hour N = minute S = second X = checksum *all time variables for this packet are in BCD form *This must be ACK'd by the PC, see below.
	or 'e'	No more records available. After this is transmitted, communication between the PC and BU is finished. (The PC does not have to ACK this in any way.)
PC -> BU	'g'	Checksum of last record was correct
	or 'b'	Checksum of last record was incorrect, please retransmit. If the BU receives this response, it should retransmit the last packet until it receives a 'g'.
	or 'e'	Tells base unit to stop sending data.

5.7 Conclusion

This chapter presented the memory configuration, user interface, schematic updates, and PC software. Much of this information is still preliminary and will be modified as necessary as the project progresses.

Chapter 6
Observations and Specifications

6.1 Introduction
Chapter 6 presents results from testing with the actual hardware. The analysis presented here includes an assessment of how well the project met specifications set at the beginning of the semester, as well as operational observations.

6.2 Overall Design
Design materials including code and circuit designs are included in the appendices.

6.3 Experimental Results
The project meets most if not all of the design specifications:

- It accurately measures the ambient temperature of the sensor. While a cheaper, more accurate and more responsive solution is a thermocouple or RTD, the IC temperature sensor was a cheap and fairly accurate solution.
- It collects a large number of temperature points under control of the computer or as a stand-alone device. An alarm sounds if any collected temperature falls outside a set range. A time stamp is stored with each temperature to aide in logging.
- It hops information across nodes if a given node is not directly within range of the base unit, hence the "ad hoc" part of this project.

Battery life and the wireless range are slightly lower than were anticipated at the beginning of the project, with the wireless range being the more worrying of the two. As is discussed in section 7.2, there are some problems with and suggestions for the wireless portion of the project.

6.4 Specifications

Wireless Range:	
Ideal conditions	~30 feet
Obstructions / noise	~10 feet
Wireless Frequency:	916.5 MHz
Wireless Modulation:	OOK (On Off Keyed)
Temperature range:	-55 °C to +125 °C
Temperature accuracy:	accurate to ± 1 °C
Max number of nodes:	127
Max number of hops:	16
Baseunit memory:	32k
Power supply:	
Baseunit	9V battery or 9V DC input
Node	2 x AA batteries
Battery Life:	
Baseunit	~72 hours
Node	~360 hours, depending on frequency of usage
Physical dimensions:	(WxLxH)
Baseunit	5 ½" x 7" x 2 ½"
Node	2 ¼" x 3" x 1 ½"

6.5 Conclusion

This project provides a useful solution for temperature logging needs. There are many applications for which this project would be suitable, and with minor modifications, this project will accommodate many more applications.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The wireless temperature network envisioned at the beginning of the semester has been designed into a hardware solution. The project works as envisioned, with some problems mentioned in the future work section of this chapter. Considering everything that has occurred in the past year, the project really has become a useful and practical solution for temperature measurement, as it was designed for in our first semester.

7.2 Future Work

Proposed future changes for this project are myriad, and so are divided into separate categories.

7.2.1 Protocol - Mobility

It was realized too late in the design that the protocol was not designed with a mobile network in mind. That is, while the protocol works well for a network in which the nodes stay in the same location, it does not handle nodes moving very well.

For instance, if a route is built to a node several hops away, that route is stored in the base unit and used indefinitely until the route breaks. This means that if the node is then moved closer to the base unit, as long as the old route is still traversable, the base unit will use it, instead of a possibly shorter and more direct new route.

A simple quick-fix might be to change the base unit software to send a route discovery packet to each node after a certain amount of time. While neither an elegant nor efficient solution, this would force the base unit to refresh the routes periodically, fixing what would otherwise be a permanent problem.

A better fix would be to redesign the node software to watch all packets, including route aware packets not destined for a node, which are usually ignored. Each node could then search the route data in the packet to see if it was a later hop in the route. If it were, then because it heard the packet before it was supposed to, a faster route must exist. The node could then send a new "route reconfigure" packet to the base unit including the new route.

7.2.2 RF Concerns

Certainly a concern for any wireless project, especially one concerned with periodic and accurate data logging, is the problem of interference from other transmitters on the same frequency. While demonstrating the project for a senior design display, many other wireless projects were nearby. Asking the nearest wireless project students, we found that their project was also using the popular unlicensed 916.5 MHz band and were transmitting periodically every 1ms! While the RFM modules are superior to the other modules considered in power consumption, a transceiver with FM capabilities would certainly have improved the noise rejection of the system.

Additionally, the wireless transceiver was connected directly to the PIC's serial input to reduce the amount of software processing necessary to successfully receive a packet. This allowed

much simpler code, but also precluded us from incorporating standard RF procedures, such as Manchester encoding or 12-bit symbol conversion to DC balance the transmission. Also, the serial port trick works poorly if the PIC misses the first few bits of a byte, but receives the later ones. It was because of this shortcut that the network was not brought to a 19200 baud transmission rate. In order to transmit at 19200 baud, the high-pass filter on the transceiver needed to be adjusted to allow the higher bandwidth signals through. This change, however, had the unexpected consequence of also allowing quite a bit of noise to the system, which had behaved quite well at 2400. Since our shortcut method is quite intolerant of noise, the network did not function well at 19200 baud and had to be brought back to 2400.

A better way to receive data from the transceiver would be to implement a software decoder which could properly sync to a signal, even if the first few bits were chopped off, and decode a DC-balanced signal. In addition, the system would be set to reject noise on the line, allowing for fast transmission rates and transmission in noisy environments.

7.2.3 Antenna – Range

While the transceivers are rated for a range up to 380 feet in ideal conditions, our experiences with them were much worse: around 30 feet when working in the lab, and around 8 feet when in the presence of other transmitters while displaying our project in the Memorial Union. While much of this problem may be associated with the shortcuts taken in the previous section, the antenna design is also probably at fault. The current design uses a standard wire as an antenna, which is cut to approximately ¼ of one wavelength. Ideally this antenna would be straight and in the same orientation as the receiving antenna, with a good ground plane below and perpendicular to it. The antenna as built, however, is curved in the case and does not have a good ground plane.

A possible solution is to increase the case size slightly to accommodate the antenna and to redesign the PCBs with a better ground plane. A better solution would be a patch antenna, which would not require a wire and can be made to take up a relatively small amount of PCB space. However, the design of a patch antenna was difficult since none of us had experience designing them.

7.2.4 Base unit expandability

The base unit needs a microprocessor with larger program memory space in order to be expanded further. Currently the base unit's program memory is 90% used, and recompiling changes in the code often results in an "out of memory" error, as the compiler has more trouble fitting code into the remaining small portions of memory. A move to the 18F series of PIC processors was considered but was rejected because a new compiler would have to have been bought. This is an option which would require almost no work and would increase the expandability of this project.

7.2.5 Data Sensitivity

The data carried by a network can be sensitive in two ways: time and content. The current project does not secure its data in either respect. Since the data in this project is time sensitive, this aspect is of particular concern. If a packet is transmitted by a node, that node does not ensure that the next node in the route gets the packet. This means that if at any point the route is

momentarily blocked, that data point in the collection will be lost. An acknowledgement system could be utilized in time sensitive applications, which would drain more battery power, but would allow nodes to ensure messages were sent. Also considered was having each node transmit to the next node, then listen for that node to transmit to its next hop. If the transmission wasn't heard, the node could retransmit, assuming its transmission was lost. Alternatively, it could send a new "route bad" packet to the base unit, speeding up delays associated with bad route timeouts on the base unit.

The harder but well documented problem of securing the content of the packets would probably use a form of encryption across the network to prevent casual sniffing of the data.

Appendix
References

A.1 References

[1] Food Safety and Inspection Service. *Transportation and Storage Requirements for Potentially Hazardous Foods.* 3 Mar 2002. Available at http://www.fsis.usda.gov/OA/background/transbkg.htm.

[2] Webopedia. *Webopedia: Network Topologies*. 3 Mar 2002. Available at http://www.webopedia.com/quick_ref/topologies.html.

[3] RL System. *IceSpy*. 28 Feb 2002. Available at http://www.icespy.com/icespyrl.htm.

[4] HIDRA. *Rockwell Scientific*. 28 Feb 2002. Available at http://www.rockwellscientific.com/hidra/index.html.

[5] MICA Motes & Sensors. *Crossbow: Smarter Sensors in Silicon*. 28 Feb 2002. Available at http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[6] TinyOS Homepage. *TinyOS: A Component-based OS for the Networked Sensor Regime*. 28 Feb 2002. Available at http://today.cs.berkeley.edu/tos/index.html.

[7] Protocols for Adaptive Mobile and Wireless Networking. *The CMU Monarch Project*. 28 Feb 2002. Available at http://www.monarch.cs.cmu.edu.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y-C. Hu and J. Jetcheva. "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols." *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM, Dallas, TX, October 1998.